8-2016

Exploring Spin-transfer-torque devices and memristors for logic and memory applications

Zoha Pajouhi
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Zoha Pajouhi

Entitled: Exploring Spin-Transfer-Torque Devices and Memristors for Logic and Memory Applications

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

KAUSHIK ROY

ANAND RAGHUNATHAN

BYUNGHOO JUNG

SAEED MOHAMMADI

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

KAUSHIK ROY

Approved by Major Professor(s): _____

Approved by: V. Balakrishnan 06/13/2016

Head of the Department Graduate Program Date

EXPLORING SPIN TRANSFER TORQUE DEVICES AND MEMRISTORS FOR LOGIC AND MEMORY APPLICATIONS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Zoha Pajouhi

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2016

Purdue University

West Lafayette, Indiana

To my Father for his endless support

To my Mother for her ceaseless advice

To my Spouse for his boundless assistance

To my Children for their blameless patience

ACKNOWLEDGMENTS

I am greatly appreciative of Prof. Kaushik Roy under whose exceptional guidance and patience I had the privilege to carry out this work. His constant support and encouragement is one of the major factors for the completion of this work. I would also like to thank my advisory committee members, Prof. Anand Raghunathan, Prof. Byunghoo Jung and Prof. Saeed Mohammadi for their constructive feedback and suggestions. I am grateful to the members of Nanoelectronic Research Lab and my friends for making my experience of working on this project memorable. Special thanks to Dr. Xuanyao Fong, for long and insightful discussions, which helped me understand the fundamentals of STT-MRAMs in a greater detail. I would also like to thank Dr. Matthew Swabey for his support during my studies. I am indebted my success to my parents Mohammad Pajouhi and Masoumeh Farahani. I would also like to thank my family, Zeinab, Hossein and Atieh Pajouhi.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

ABSTRACT

Pajouhi, Zoha, Ph.D. Purdue University, August 2016. Exploring Spin-transfer-torque devices and memristors for logic and memory applications. Major professor: Kaushik Roy

As scaling CMOS devices is approaching its physical limits, researchers have begun exploring newer devices and architectures to replace CMOS.

Due to their non-volatility and high density, Spin Transfer Torque (STT) devices are among the most prominent candidates for logic and memory applications. In this research, we first considered a new logic style called All Spin Logic (ASL). Despite its advantages, ASL consumes a large amount of static power; thus, several optimizations can be performed to address this issue. We developed a systematic methodology to perform the optimizations to ensure stable operation of ASL.

Second, we investigated reliable design of STT-MRAM bit-cells and addressed the conflicting read and write requirements, which results in overdesign of the bit-cells. Further, a Device/Circuit/Architecture co-design framework was developed to optimize the STT-MRAM devices by exploring the design space through jointly considering yield enhancement techniques at different levels of abstraction.

Recent advancements in the development of memristive devices have opened new opportunities for hardware implementation of non-Boolean computing. To this end, the suitability of memristive devices for swarm intelligence algorithms has enabled researchers to solve a maze in hardware. In this research, we utilized swarm intelligence of memristive networks to perform image edge detection. First, we proposed a hardware-friendly algorithm for image edge detection based on ant colony. Next, we designed the image edge detection algorithm using memristive networks.

1. INTRODUCTION

As CMOS scaling benefits encounters huge reductions due to reduced feature size, there is an ongoing research to identify newer technologies to replace CMOS. According to the International Technology Roadmap for Semiconductors [1] (ITRS) their scaling will be continued over the next several years. However, the devices will reach the fundamental physical limits [2].This fact has propelled research on new devices and logic styles that can potentially replace CMOS. Even though the quest for new devices may not lead to an effective replacement for CMOS, new research has opened the door for new devices that can potentially be effective in applications where CMOS is not an ideal option. Among the most promising beyond CMOS devices are Spin Transfer Torque (STT) devices. The research on STT devices was initialized when Slonczewski and Berger [3,4] projected that the current flowing through a metallic layer can generate spin transfer torque strong enough to reorient the magnetization in one of the layers [5]. Ever since this phenomenon was discovered, different structures have been proposed that utilize STT switching for different computing purposes [6,7,8].

One of the most well known STT devices is the Magnetic Tunnel Junction (MTJ). The MTJ consists of a free and a fixed ferromagnetic layer as well as an insulator. The insulator is sandwiched between the two ferromagnetic layers. If there is a current flow through the MTJ, spin polarized carriers tunnel through the insulator and insert a torque on the free layer that can reorient its magnetization. MTJs are utilized in different spintronic structures such as Spin Transfer Torque – Magnetoresistive Random Access Memories (STT-MRAMs). Due to their non-volatility, unlimited endurance and the capability to be integrated with CMOS, STT-MRAMs are considered as one of the candidates for future universal memories. An STT-MRAM bit-cell consists of an MTJ and an access transistor, which is used to access the MTJ. The transistor is used for both read and write operations. This dual-purpose utilization of the access transistor imposes

important restrictions in the STT-MRAM bit-cell design. These restrictions are due to conflicting requirements for read and write operations, which lead to higher bit-cell probability of failure and yield degradation. Research has begun in earnest to analyze STT-MRAMs from reliability point of view and to mitigate the high probability of failure associated with STT-MRAMs [9,10,11]. However, the approaches taken so far are either focused on failure enhancement at the bit-cell level [12,13,14] or on memory architecture design techniques [15,16]. Since these solutions only consider a specific level of design abstraction, they lead to significant overdesign resulting in increased power dissipation. In this research, we proposed a unified device-circuit-architecture co-design simulation framework for yield enhancement. We considered device parameters and bit-cell level parameters together with diverse ECC schemes to streamline the robustness and energy efficiency of the STT-MRAM cache.

Furthermore, we proposed usage of Failure aware ECC (FaECC) for STT-MRAMs. In FaECC, we enhanced the correction capability of ECC to mask hard errors while maintaining its capability to correct soft errors. Eventually, we took advantage of this enhanced correction capability to ease the strict design requirements of STT-MRAMs in terms of area, energy or performance.

We also considered STT devices for logic applications. For this purpose, we analyzed All Spin Logic (ASL) which is a logic style based on Lateral Spin Valve (LSV) [7]. The lateral spin valve is a three terminal device in which charge current is passed through two of its terminals. The spin-polarized current is transferred to the third terminal due to spin diffusion. If the device is properly designed, this diffused spin-polarized current exerts a torque on a ferromagnetic layer placed in the third terminal and can potentially change its magnetization orientation. The LSV structure can be modified to build a new Boolean logic style such as ASL [24,25]. ASL benefits include high density, non-volatility and low operating voltage. However, with current achievable device parameters, it suffers from low speed and large static power consumption. In order to make ASL more efficient, there are several optimizations that can be performed such as power gating, pipelining and stacking [17]. We propose a systematic methodology to automatically utilize these optimizations on a wide range of benchmarks. Furthermore,

we utilized the proposed methodology to evaluate different logic styles and analyzed the suitability of ASL for different categories of logic applications.

We also considered the unique characteristics of other beyond CMOS devices for non-Boolean computing. To this end, we considered the similarities between memristive devices and swarm intelligence to perform image edge detection. Specifically, we proposed a hardware friendly algorithm based on swarm intelligence for image edge detection. We designed the proposed algorithm using state-of-the-art memristive devices.

This thesis is organized as follows. In Chapter 2, we propose a systematic methodology for ASL evaluation. In Chapter 3, we go over STT-MRAM fundamentals and different failure mechanisms in STT-MRAMs. In Chapter 4, we propose a device/circuit/architecture framework for reliable design of STT-MRAMs. In Chapter 5, we propose failure aware ECC and bit-cell design method for STT-MRAM bit-cells. In Chapter 6, we propose usage of memristive networks based on swarm intelligence for image edge detection. Finally, in Chapter 7 we conclude the thesis and explain potential grounds for future work.

2. SYSTEMATIC METHODOLOGY FOR ASL EVALUATION

As CMOS devices scale down to the deep nanometer regime and approach their fundamental physical limits, the traditional benefits in power and performance associated with technology scaling have subsided due to increased short-channel effects and leakage power consumption [1]. This has motivated researchers to explore newer devices that can potentially replace CMOS as the next Boolean "switch" [18,19]. Specifically, recent advances and experiments [20,21,22,23,24,25,26] on spin-transfer-torque (STT) devices have identified the possibility of using "spin" (rather than "charge") as the state variable for computation. STT devices manipulate the spin orientation of a nanomagnet using a spin-polarized current to switch between Boolean logic states. These devices possess several desirable characteristics: (i) they are *non-volatile*, since the spin orientation is retained in the nanomagnet even when the power supply is turned off, (ii) they offer high *density*, and (iii) they can be operated at very low voltages in the order of 10 mV. Due to these characteristics, STT devices have been extensively explored and demonstrated to be efficient in the design of both on-chip and off-chip memories.

The intrinsic benefits of STT devices can also be potentially leveraged in the context of logic design. Towards this end, All Spin Logic (ASL) is a recently proposed approach to implement Boolean logic functions using spin-based devices. ASL circuits are comprised of a network of nanomagnets, which represent the internal logic signals of the circuit, interconnected through non-magnetic metallic channels. We illustrate the principle and operation of ASL through the example shown in Fig. 2.1 (a). The circuit consists of 2 nanomagnets - an injecting or input nanomagnet (*M1)* and receiving or output nanomagnet (*M2)*- connected through a spin channel. This 1-input 1-output circuit can operate as either a buffer or an inverter as explained below. Let us assume that the nanomagnets are initially polarized in opposite directions - *M1* to the right and *M2* to the

left. Now, when a current is passed from top to bottom through *M1,* the nanomagnet acts as a *polarizer* and polarizes the spin of the electrons parallel to its orientation (right-spin in this case). This spin-polarized current travels through the channel and exerts a spin torque on *M2* based on the principle of non-local spin torque (NLS). If the exerted spin torque is strong enough, the orientation of *M2* is reversed to align with *M1*. Thus, by passing a current through *M1*, its spin orientation is transferred to *M2*, thereby realizing the functionality of an inverter. Interestingly, the circuit can also operate as a buffer if the direction of current through *M1* is reversed *i.e.,* the current is passed from bottom to top as shown in Fig 2.1 (b). In this case, *M1* polarizes the current in a direction opposite to its orientation resulting in the channel current, and hence *M2,* to orient opposite to *M1*. Thus the circuit can either "copy" or "invert" the orientation of the input (*M1)* to the output (*M2)* based on the direction of current through *M1*.

Recent efforts [24,26] have adopted a similar approach to design complex ASL gates such as AND, OR, XOR *etc*. Also, the design of a compact full-adder using ASL has been proposed [27], which has in turn been used to design and evaluate an ASL implementation of the Discrete Cosine Transform (DCT). While the above efforts have made a promising start, the suitability of ASL or the realization of larger and a broader range of logic circuits remains unexplored. To enable such an exploration, we propose a systematic methodology to synthesize arbitrary logic circuits using ASL. Our methodology incorporates various optimizations that exploit the unique properties of ASL. First, we exploit the non-volatility of nanomagnets to realize storage elements with minimal cost. Second, power is consumed in the all-metallic ASL gates regardless of whether or when any useful switching occurs. Thus, power gating or "clocking" ASL gates in a fine-grained manner is critical to avoid significant energy penalties due to leakage. However, power gating incurs area and energy overheads in the form of gating transistors, thus requiring proper analysis and optimization. Towards this end, our methodology automatically identifies ASL gates with similar time-periods of evaluation and clusters them into gating domains. ASL gates within each gating domain share a gating transistor, thereby amortizing the overheads of these transistors while retaining a significant fraction of the energy benefits.

Another avenue for energy optimization in ASL circuits is “magnet stacking” [17], which refers to connecting the terminals of multiple nanomagnets in series. Through manual design of arithmetic circuits, stacking has been shown to significantly improve the energy efficiency of ASL. Finally, a key consideration in designing ASL circuits is that, since ASL gates communication using spin-current, the physical length of interconnects cannot exceed the so-called spin diffusion length or spin flip length of the communication channel. Thus, buffers should be inserted in larger interconnects to ensure correct functionality. Finally, taking advantage of the built-in non-volatility, ASL circuits can be operated in a pipelined fashion, improving their throughput.In summary, the key contributions of this work are:

- We propose the first systematic design methodology for implementing arbitrary logic circuits using ASL. Given an RTL description, the proposed methodology synthesizes an optimized ASL implementation of the circuit.
- The proposed methodology incorporates optimizations such as intra-cycle power gating, fine-grained logic pipelining, and nanomagnet stacking, which exploit the unique properties of STT devices.
- We utilize the design methodology to evaluate ASL across a broad range of designs, including combinational and sequential logic benchmarks, DSP circuits, and the Leon SPARC3 general-purpose processor.

Based on our evaluation, we draw key conclusions about the viability and competitiveness of ASL for different performance scenarios. We also evaluate the impact of potential improvements in key material and device parameters on the efficiency of ASL.

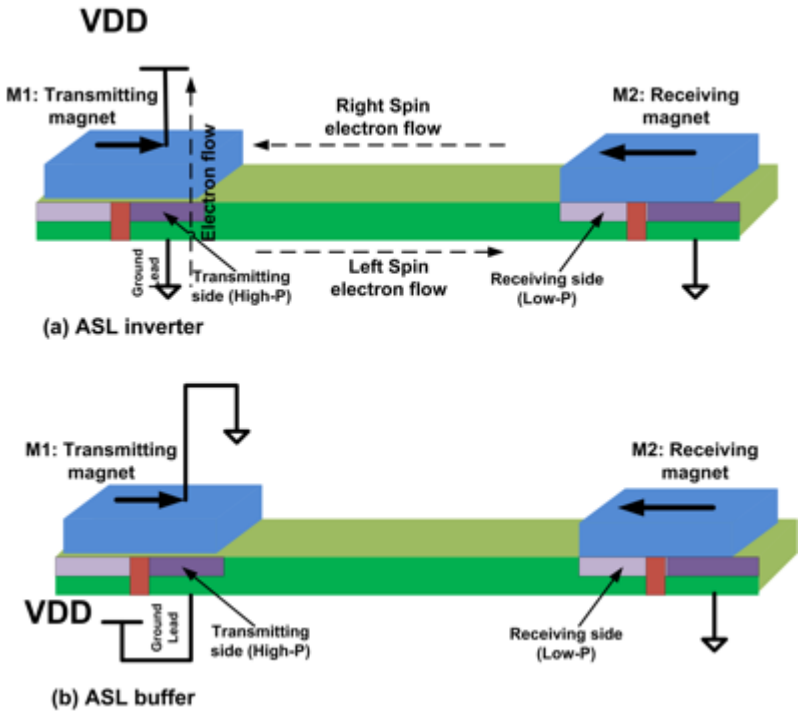


Fig. 2.1. (a) Schematic of an ASL inverter and (b) Schematic of an ASL buffer.

2.1. All Spin Logic: Preliminaries

All Spin Logic operates on the basic principle of storing state information in the spin of electrons (magnets) and manipulating the state using spin-polarized currents. An ASL buffer (inverter) consists of two nanomagnets that are connected through a non-magnetic and metallic channel as shown in Fig. 2.1 In order to understand the operation of the buffer (inverter), we will distinguish between "charge current" and "spin current". The input nanomagnet, *M1*, (connected to a supply voltage) injects spin-polarized electrons into the channel (which is made of a non-magnetic material such as Cu). Even though the charge current flows from supply to ground, spin-polarized current can flow through the channel due to the spin potential difference across it. The spin current exerts a torque to flip the output nanomagnet, *M2*. Note that, in Fig. 2.1 (a) the left-spin electrons flow to the right and right-spin to the left in the channel, making the total charge current to be zero. Since the channel is non-magnetic, the spin orientations of electrons would get randomized beyond the spin diffusion length. For Cu, the spin flip or spin diffusion length is about 500nm [42]. In other words, if the input and output nanomagnets are separated by more than 500nm, either there is a need for repeaters, or it

will not be possible to switch the output nanomagnet, *M2*, using spin-polarized current injected from nanomagnet *M1*. The high and low polarized layers placed below the nanomagnets guarantee the directionality of the logic gate.

In order to simulate an ASL gate, each of the gate's elements such as the nanomagnets and the channel are modeled as four component conductance elements: one charge conductance and three spin conductances [24,28,29]. Then, the current is derived and inserted into a magnetization solver. The magnetization dynamics are determined by self-consistently solving the current equation using the spin conductances and the Landau-Lifshitz-Gilbert (LLG) equation [29]. A more detailed description of the simulation methodology can be found in Section 2.3.

The basic ASL inverter can be extended into a majority gate as shown in Fig. 2.2, where *A*, *B* and *C* are the input nanomagnets and *D* is the output nanomagnet. It is well known that majority gates can be used to realize other logic functions. For example, if one of the nanomagnets in Fig. 2.2 is fixed to logic `1', the gate computes the OR of the remaining inputs. It is of particular interest that each nanomagnet can be considered as a storage element. Hence, if properly designed, there is no need for separate flip-flops or latches in ASL circuits.

Since ASL devices are all metallic, they are capable of functioning at very low voltages (~10mV). However, they have a poor performance in comparison with CMOS. Improved performance can be achieved by injecting larger current through the nanomagnets. In other words, the operating frequency can be improved by increasing the supply voltage. However, increasing the current has a detrimental effect on the power consumption (much larger short-circuit or direct path current since the devices are all metallic) and can cause reliability concerns.

Unlike CMOS circuits in which power consumption is higher when it switches and lower at other times, ASL circuits consume similar power regardless of whether they are switching (due to the metallic direct path from supply to ground). Therefore, there is a need for "power gating" or "clocking" the devices with an added transistor to reduce power dissipation when the gate is not evaluating. Fig. 2.3 depicts two cascaded gates with power gating. The transistors are used for powering the ASL nanomagnets. For

example, *M1* is enabled through *T1* and if *T1* is turned on, inverter 1 will be evaluated and *M2* flips according to its input. At the next step, *M1* can be switched off and *M2* can be turned on; thus, inverter 1 is turned off and inverter 2 is turned on. At this step, the new magnetization of *M2* gets propagated to *M3*.

The ASL gate can be viewed as a resistor whose resistance is based on the size of the nanomagnet (typically a few Ohms) in series with the MOSFET. The voltage across the source to drain of the transistor can be of the order of few tens of millivolts, mainly determined by the energy restrictions (note that the gating transistor operates in the triode region). However, since the resistance of the triode region transistor is expected to be much larger (in the range of KOhms) than the resistance of the nanomagnet (few Ohms), most of the power dissipation will be in the gating transistor. In order to mitigate this drawback, one can stack several nanomagnets and share one gating transistor for all of them (note that this can only be done when the nanomagnets are evaluated simultaneously). This enhances power efficiency because the power consumption of the MOSFET is amortized across the nanomagnets sharing that transistor. Fig. 2.4 shows a sample circuit with stacking. It can be observed that since *A*, *B* and *C* are inputs to the same gate and are active simultaneously, T1 can be used to power gate all nanomagnets corresponding to *A*, *B* and *C*.

It is evident from the above discussions that in order to evaluate the potential of ASL, there is a need to utilize suitable design techniques including power gating and stacking of ASL gates. In addition, as described in the next section, the performance of ASL circuits can be significantly improved through fine-grained pipelining by exploiting the built-in sequential elements in ASL gates. We next describe a synthesis methodology for ASL that incorporates these optimizations.

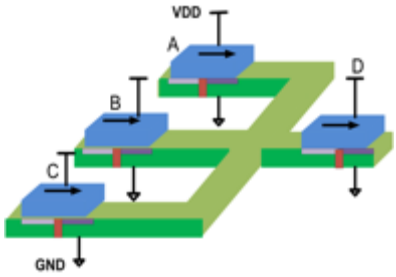


Fig. 2.2. ASL majority gate circuit.

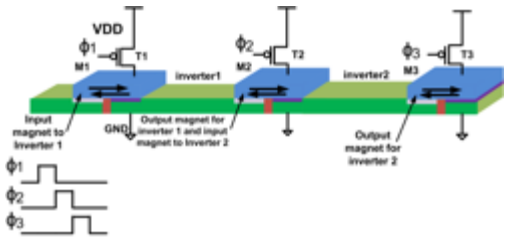


Fig. 2.3. ASL power gating circuit.

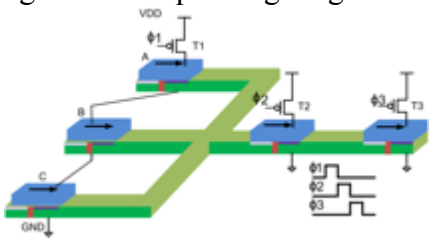


Fig. 2.4. Sharing/stacking of nanomagnets.

2.2. ASL Synthesis methodology

In this section, we explain the proposed synthesis methodology for ASL. The methodology takes an RTL description of the circuit as its input and produces an ASL implementation (ASL gate netlist with the required gating transistors) that is optimized for energy. Fig. 2.5 shows the various steps involved in the synthesis process. First, a standard cell technology library is developed for ASL, in which different logic gates are optimized and characterized using a physics-based simulation framework. Next, a commercial logic synthesis tool is utilized to perform technology-independent optimization and a basic mapping of the circuit to the ASL technology library. Several design optimizations that leverage the unique characteristics of ASL nanomagnets are then applied to enhance the energy-efficiency of the synthesized netlist. These include: (i) *Intra-cycle power gating*, in which the nanomagnets in the ASL netlist are clustered into multiple gating domains, each of which are dynamically power gated during circuit operation, (ii) *Nanomagnet stacking,* in which the nanomagnets within each gating domain are further sub-divided and their terminals are "stacked" together in series, and (iii) *Fine grained logic pipelining*, in which the logic paths within each gating domain are balanced through the insertion of buffers and are operated in a pipelined fashion. We note that the above optimizations incur overheads in terms of both area and energy and hence require careful analysis of the trade-offs involved. In addition, to ensure correct

operation, our synthesis methodology also performs fanout optimization and approximate placement followed by interconnect buffer insertion. The following subsections provide a detailed description of the above steps and the various heuristics employed in their implementation.

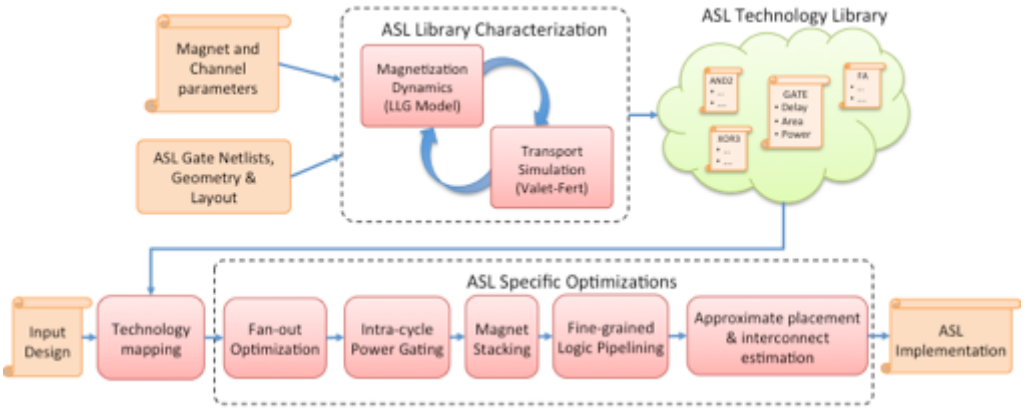


Fig. 2.5. ASL design methodology.

2.2.1. ASL technology library generation

The ASL technology library comprising of a range of logic gates with varying energy-delay characteristics is developed using a rigorous physics-based simulation framework. A physics-based simulation framework for ASL analysis was proposed in [24,29]. In order to simulate ASL devices and to determine the switching speed and energy consumption of ASL gates, the *spin transport* through the devices and the *magnetization dynamics* have to be solved self-consistently. The transport simulation is based on a modified Valet-Fert model [30] and the magnetization dynamics are computed using the Landau-Lifshitz-Gilbert (LLG) equation with spin-torque [31]. The transport simulation model receives the magnetization vector ($\hat{m} = M_x, M_y, M_z$) of the nanomagnets and calculates the current in the channel as a 4-component vector ($I_{4x1} = \left[I_c, I_s^x, I_s^y, I_s^z\right]$). The channel current is then fed to the magnetization dynamics model, which provides the magnetization vector as the output. Thus, the magnetization dynamics is a function of spin currents, which in turn, are estimated based on the spin transport simulations. On the other hand, the transport calculations depend upon the direction of spin momentum in the nanomagnet and the spin channel. Hence the models

are solved self-consistently to accurately estimate the switching time and energy of ASL gates.

In this work, we adopted the modular approach developed in [17,29] to simulate the ASL gates. We explain the different steps involved in the process using a buffer shown in Fig 2.6. From a modeling perspective, the ASL device can be divided into 4 regions as shown in Fig. 2.6 (a): (a) non-magnetic channel to transport the charge current from the contact lead to the nanomagnet, (b) ferromagnet that acts as a source of spin polarized electrons, (c) an interface region between ferromagnet and non-magnetic channel that enhances the injection of spins from the nanomagnet into the channel and (d) non-magnetic channel to transport spin current from the input nanomagnet to the output. The connections between these four regions are shown in Fig 2.6 (b). Each of these 4 regions is modeled using a lumped π-conductance model with a series element (G_{se}) and two shunt elements (G_{sh}). Since spin current is modeled as a vector, all the nodal voltages and branch currents of the circuit are represented using four components ([Vc,Vz,Vx,Vy] and [Ic,Iz,Ix,Iy]) – one element for charge and 3 directional elements for spin. All the conductances (G) are represented by 4x4 matrices which connect the branch currents with nodal voltages as per Ohm's law:

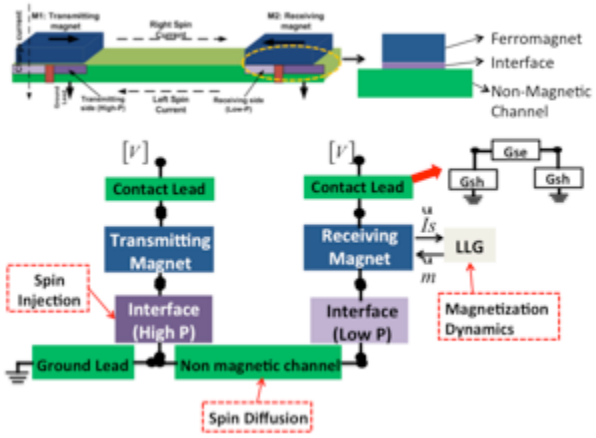


Fig. 2.6. The different building blocks of the ASL

$$\bar{I} = \left[I_c \; I_{sx} \; I_{sy} \; I_{sz}\right]^T \tag{2.1}$$

2.2.1.1 Non-magnetic channel

The spin current from transmitting nanomagnet diffuses to the receiving nanomagnet through non-magnetic (NM) channel. The lumped series (Gse_ch) and shunt (Gsh_ch) conductance elements of non-magnetic channel are as represented below.

$$G_{se\_ch} = \begin{bmatrix} \frac{A_{ch}}{\rho_{ch} L_C} & 0 & 0 & 0 \\ 0 & \frac{A_{ch}}{\rho_{ch}\lambda_s}\operatorname{csch}\left(\frac{L_C}{\lambda_s}\right) & 0 & 0 \\ 0 & 0 & \frac{A_{ch}}{\rho_{ch}\lambda_s}\operatorname{csch}\left(\frac{L_C}{\lambda_s}\right) & 0 \\ 0 & 0 & 0 & \frac{A_{ch}}{\rho_{ch}\lambda_s}\operatorname{csch}\left(\frac{L_C}{\lambda_s}\right) \end{bmatrix} \quad (2.2)$$

$$G_{sh\_ch} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{A_{ch}}{\rho_{ch}\lambda_s}\operatorname{csch}\left(\frac{L_C}{2\lambda_s}\right) & 0 & 0 \\ 0 & 0 & \frac{A_{ch}}{\rho_{ch}\lambda_s}\operatorname{csch}\left(\frac{L_C}{2\lambda_s}\right) & 0 \\ 0 & 0 & 0 & \frac{A_{ch}}{\rho_{ch}\lambda_s}\operatorname{csch}\left(\frac{L_C}{2\lambda_s}\right) \end{bmatrix} \quad (2.3)$$

Here, L_C is the length of the channel, A_{ch} is the area of the cross section, ρ_{ch} is the resistivity and λ_s is the spin diffusion length. The first element of the series component is the charge conductance arising from ohm's law and the other elements are spin diffusion terms. Note that the first element of the shunt component is zero indicating that the shunt component only acts as a spin sink.

2.2.1.2 Ferromagnetic (bulk) region

The four component lumped conductance matrices for a nanomagnet aligned along the 'z' direction is given by:

$$G_{se\_FM} = \frac{A}{\rho L}\begin{bmatrix} 1 & P & 0 & 0 \\ P & P^2 + \alpha\,\operatorname{csch}(\frac{L}{\lambda_s}) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \alpha = (1-P^2)(\frac{L}{\lambda_s}) \quad (2.4)$$

$$G_{sh\_FM} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{A}{\rho L} + \alpha\tanh(\frac{L}{2\lambda_s}) & 0 & 0 \\ 0 & 0 & g'_{sf} & 0 \\ 0 & 0 & 0 & g'_{sf} \end{bmatrix} \quad (2.5)$$

Where P is the spin polarization factor.

2.2.1.3 Channel-magnet interface

The series and shunt components of the interface region consider the mode mismatch between ferromagnet and the non-magnetic channel.

$$G_{se\_int} = \frac{q^2}{h} M \begin{bmatrix} 1 & P & 0 & 0 \\ P & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.6}$$

$$G_{sh\_int} = \frac{q^2}{h} M \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -b & a \end{bmatrix} \tag{2.7}$$

For ohmic contacts a~1 and b~0. For circuit simulation, the ASL device is divided into different building blocks/components connected to each other with each component represented by a lumped π conductance model. The magnetization dynamics of the output nanomagnet is obtained by solving the Landau-Lifshitz-Gilbert (LLG) equation [32]:

$$(1+\alpha^2)\frac{d\hat{m}}{dt} = -\gamma\left(\hat{m}\times\vec{H}\right) - \alpha\gamma\left(\hat{m}\times\hat{m}\times\vec{H}\right) + \vec{\tau} + \alpha(\hat{m}\times\vec{\tau}), \tau = \frac{\hat{m}\times\vec{I_S}\times\hat{m}}{qN_S} \tag{2.8}$$

Here α is the Gilbert damping constant, $\hat{m}$ is the magnetization of the receiving or output nanomagnet, γ is the gyromagnetic ratio, q is the charge of an electron, *Ns* is the total number of spins in the nanomagnet given by the relation:

$$N_s = \frac{M_S \Omega}{\mu_B} \tag{2.9}$$

(M_S - Saturation magnetization, Ω-Volume of the nanomagnet and μ_B is the Bohr magneton). $\vec{H}$ is the effective field which represents the sum of internal and external fields on the nanomagnet. Fig. 2.7 shows the magnetization dynamics of 'z' component for I_{in}=500μA and I_{in}=5mA. As expected the nanomagnet can be switched faster with higher input current; however, it leads to higher power dissipation.

Utilizing the simulation framework described above, ASL gates of different functionality can be designed and characterized. The ASL library developed in this work contains 30 logic gates with a maximum fan-in size of 7. There are several design parameters that affect the delay and energy characteristics of ASL gate. The first parameter is the ground resistance. In ASL gates, only a portion of the input spin current flows through the spin channel to the output nanomagnet, while the rest of it is lost to the

ground. The fraction of the spin current that flows to the ground can be manipulated by varying the ground lead resistance (R_g) of the gate. Thus, the ground resistance has a direct impact on the delay characteristics of the ASL gate. As shown in Fig. 2.8, as the ground resistance is increased, a larger fraction of input spin current reaches the output nanomagnet and hence the delay of the gate decreases. Note that the charge component of the input current completely flows to the ground as the output nanomagnet in the ASL gate is floating. As described in Section 2.1, fine-grained power gating is key to the energy efficiency of ASL. In order to power gate the device, one has to connect it to a transistor. The gating transistor should be designed to supply the required current (I_{ON}) during the evaluation period. For proper operation, the gating transistors require a substantially higher power supply in comparison with the power supply required for ASL gate operation. This large power supply is due to the R_{ds} of the transistor. However, upsizing the transistor increases its dynamic power. Therefore, there is a trade-off between the dynamic and static power overhead of the gating transistor. Fig. 2.9 shows the static and dynamic power consumption of an inverter versus the normalized area of the transistors for a constant I_{ON}. It can be observed that the static power decreases drastically and the dynamic power increases with an increase in the area of the transistor.

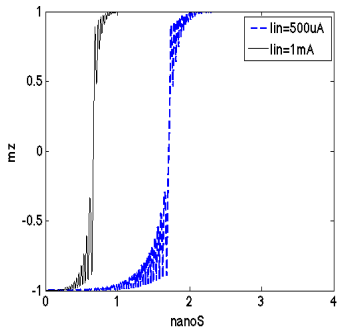


Fig. 2.7. Magnetization dynamics of ASL inverter for Iin=500μA and 1mA.

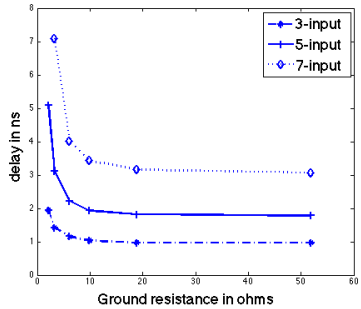


Fig. 2.8. Switching delay *vs.* ground resistance for different ASL gates.

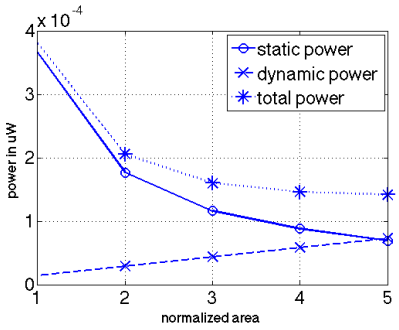


Fig. 2.9. Power *vs*. relative area for an inverter.

2.2.2. Technology mapping and fan-out optimizations

Once the ASL library is generated, utilize a commercial synthesis tool (e.g. Synopsys Design Compiler [33]) to map the input RTL to the ASL library.

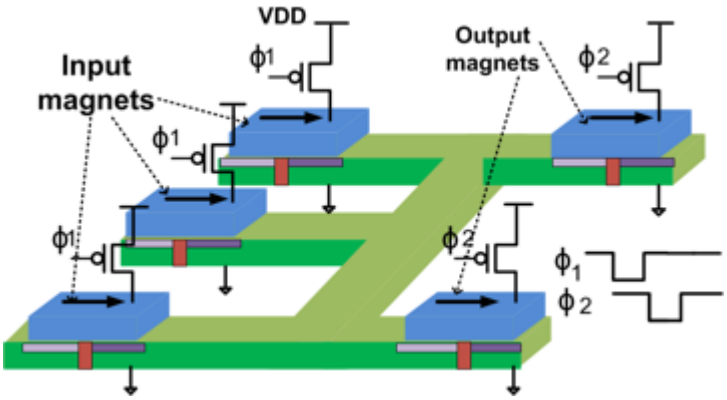
While this yields a functional map to the standard cells in the ASL library, additional optimizations are required to ensure correct operation due to the following circuit-level considerations: (i) The fan-out of ASL gates cannot exceed 1. This is because the spin current in the channel is sufficient to influence only one receiving or output nanomagnet. (ii) The length of interconnects or spin channels between successive gates cannot exceed the spin diffusion length of the material.

We propose two different methods *viz*. performance optimized fan-out and power optimized fan-out, to achieve proper operation for gates with larger fan-outs. In the case of performance-optimized fan-out, as shown in Fig 2.10 (a), the number of receiving nanomagnets is increased to the number of fan-outs of the gate (2 in this example). However, the gating transistors of all the input nanomagnets are sized up such that the current through the spin channel is sufficient enough to flip all the receiving nanomagnets. Clearly, this incurs area and power overheads, but does not result in any performance degradation.

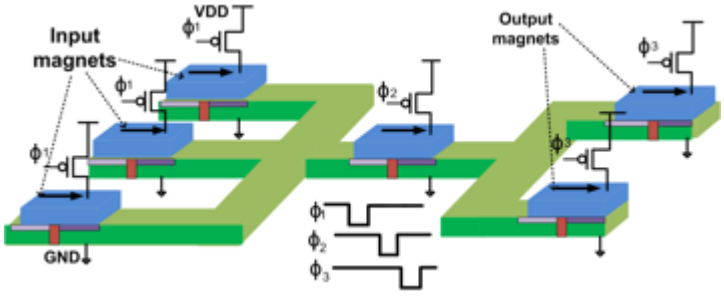
On the other hand, as shown in Fig 2.10 (b), the power-optimized fan-out evaluates the gate in two stages. In the first stage, the logic is evaluated by supplying current through the input nanomagnets and the output is stored in an intermediate receiving nanomagnet. In the second stage, a larger current, sufficient to flip all the output nanomagnets, is passed through the intermediate nanomagnet and its spin orientation is transferred to the outputs. Since power-optimized fan-out introduces an

extra stage of logic, it potentially degrades performance if the gate lies in the critical path. However, it alleviates power overheads, as a larger current is only supplied to the intermediate nanomagnet, as opposed to all the input nanomagnets in the previous case. In our methodology, we perform power-optimized fan-out if there exists timing slack at the output of the gate; else we resort to performance-optimized fan-out.

The above steps provide a fully functional ASL implementation of the given circuit. Next, we perform several optimizations to improve the energy consumption of the circuit, which are detailed in the following subsections. We illustrate different optimizations using an example circuit shown in Fig 2.11 The corresponding ASL gate netlist obtained after technology mapping including power-optimized fan-out is shown in Fig. 2.12. Note that M4-M6 in Fig. 2.12 are fixed nanomagnets used to realize different logic functions from majority gates.



(a)



(b)

Fig. 2.10. (a) Performance optimized fan-out. (b) Power optimized fan-out.

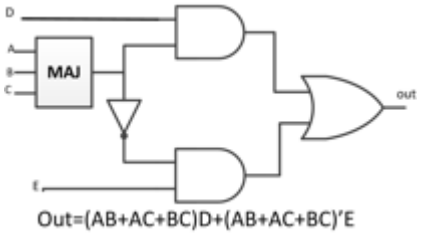


Fig. 2.11. Illustration-Sample circuit-gate level implementation.

2.2.3. Generation of power gating domains

As noted earlier, one of the key differences between CMOS and ASL is that the energy consumption of the circuit is not data-dependent, *i.e.,* energy is consumed when current is supplied to the input nanomagnets of a gate, regardless of whether the output nanomagnet changes its spin orientation. Hence, it is critical to turn-off the current supply (by switching-off the gating transistor) to the input nanomagnets as soon as the evaluation of the gate is complete. This is possible because the nanomagnets are non-volatile and they retain their state even when the gating transistor is turned off.

Power gating each gate individually will incur significant overheads in terms of the logic required to generate the power gating signals. Instead, we cluster the gates in the circuit into several "gating domains" and gates within each domain are evaluated and power-gated together. Now, each gating domain should be powered ON for the duration equal to the union of the evaluation periods of constituent gates. Clearly, this duration is greater compared to when each gate was power-gated individually. Therefore, while clustering reduces the energy overheads associated with generating the power-gating signals, it negatively impacts the evaluation energy of the gating domains, as the gates are potentially powered ON for a longer period of time. An extreme example is when all gates in the circuit are clustered to a single gating domain, in which case they are powered ON for the entire gating period resulting in significant energy penalties. Thus, the circuit should be carefully clustered such that the number of gating domains is minimized. On the other hand, while ensuring the evaluation period of the gates is minimum. Fig. 2.13 shows the pseudo-code used to generate the gating domains. The algorithm is based on searching to obtain a local minimum for the energy consumption [34]. The algorithm takes an input gate netlist and the pulse width of the multiphase clock

gating as its inputs and produces a clustered netlist with the logic necessary to power-gate the gating transistors. First, the circuit is levelized and the gates are sorted in topological order. Next, gates within the same logic level are clustered together to form separate gating domains. The intuition behind this clustering is that gates in the same level of logic have similar arrival times. This initial cluster configuration is further improved such that the overall energy is minimized. We specifically consider the critical gates in each cluster, *i.e.,* the gates at the boundaries of the cluster that determine the time duration for which the cluster should be powered ON. We perturb the cluster configuration by moving the critical gates between clusters. For example, consider the case where gate *G* originally in cluster *i* is moved to cluster *j*. This results in the ON duration of cluster *i* reducing by ΔT_D and potentially increases the ON duration of cluster by *j* by ΔT_I. Also, let N_i and N_j be the number of gates in the clusters *i, j* respectively. Then, we estimate relative energy savings when the gate is transferred to be the difference between the product of the number of gates in the cluster and the change in its time duration ($\Delta T_I . N_j$ - $\Delta T_D . N_i$). If this value is beyond a threshold, then the gate is transferred from cluster *i* to *j*. This process is iterated until no gate movement changes the clusters and the final cluster configuration is thus obtained.

Given the cluster configuration, the power gating signals for each cluster are generated using a multi-phase clock. The clusters are sorted in topological order and based on the time duration, for which they are required to be powered ON, appropriate number of gating phases are assigned. Thus intra-cycle power gating is achieved. Fig. 2.14 shows the ASL implementation of the sample benchmark with power gating transistors added. In this case, the sample benchmark has 5 gating domains for the different levels of logic in the circuit. As seen, A,B,C are assigned to the gating domain 1, M1 is assigned to gating domain 2, D,E and M2-M5 are assigned to domain 3, M6-M8 are assigned to domain 4 and finally the output nanomagnet is assigned to domain 5.

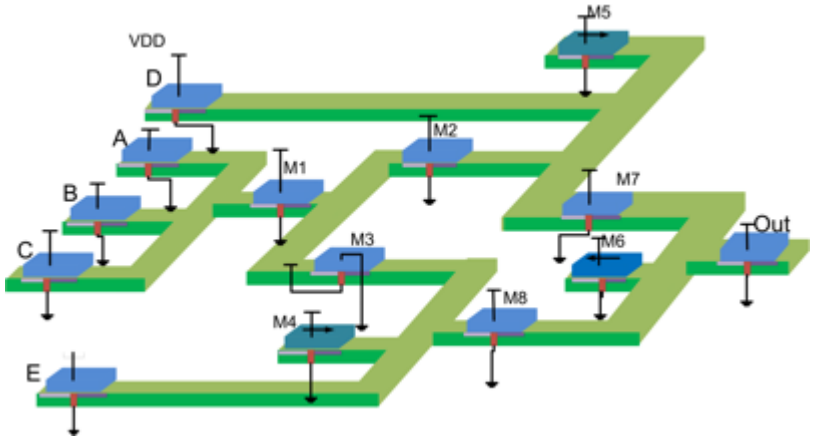


Fig. 2.12. Illustration-Sample circuit-gate netlist after technology mapping

Algorithm 1 Generate Gating Domains for Intra-cycle Power Gating

Input: Circuit (CKT), Multi-phase Gating
Output: Gating domains {1,2,3, … K}
Gates in each gating domain {C_1, C_2, … C_K}

1. Levelize CKT
2. Assign gates in Level *i* → clock domain C_i
3. {N_1, N_2, … N_K} = Number of gates in each domain
4. $\Delta M = \infty$
5. **While** ($\Delta M > 0$) {
6. $\Delta M = 0$
7. **For** i = 1 to K % each clock domain
8. G_S = {Gates that determine *start* time of domain i}
9. G_E = {Gates that determine *end* time of domain i}
10. ΔT_D = Time decrease of i when G_S moved to i-1
11. ΔT_I = Time increase of i-1 when G_S moved from i
12. **If** ($\Delta T_I . N_{i-1} - \Delta T_D . N_i$) {
13. Move G_S → domain i-1
14. ΔM += G_S
15. N_i -= G_S ; N_{i-1} += G_S
16. } % End if
17. Repeat lines 10-15 by moving G_E to domain i+1
18. **If** (N_i == 0) % if all gates moved to adjacent domains
19. Delete domain; K -= 1
20. } % End for
21. } % End while
22. No of Clock phases = K
23. Assign clock domain *i* to clock phase i

Fig. 2.13. Power gating algorithm.

2.2.4. Nanomagnet Stacking

Another key avenue for energy optimization in ASL circuits is to share the gating transistors amongst gates within a gating domain, since they are evaluated and power-gated together during circuit operation. At the circuit level, each nanomagnet can be modeled as a resistor, with resistances as low as 2Ω to 30Ω. The key idea is to connect the nanomagnets (both within and across gates) in series with the gating transistor. This arrangement is referred to as "stacking". Stacking nanomagnets results in significant power improvements as follows. Let us assume the simple case of gating without any stacking involved. Under this assumption, the number of gating transistors would be equal to the number of nanomagnets. Let V_A be the operating voltage of the ASL nanomagnets and V_C denote the voltage drop in the transistor. Then, a first order approximation of the static power *(P*) consumed by the ASL circuit is given by Equation 2.10.

$$P = (V_C + V_A).\ N.\ I \tag{2.10}$$

In the above equation, *N* represents the number of nanomagnets (or gating transistors) in the circuit and *I* denotes the current. In the ideal case of $V_C \approx 0$ there is no need for stacking because the overhead of the transistor is negligible. However, due to the ON resistance of the gating transistors, the overhead of the transistors is large. Therefore, $V_C >> V_A$ and hence, the power consumption of the circuit is dominated by the gating transistor.

Now, let us assume that the network topology allows stacking of '*m*' nanomagnets. In this case, the power consumed (P_{STACK}) is given by Equation 2.11.

$$P_{STACK} = (V_C + m.V_A)\ .\ (N/m)\ .\ I \tag{2.11}$$

Since the nanomagnets are connected in series, the supply voltage is increased to $V_C + m.V_A$. However, the relative increase in the supply voltage is insignificant for small values of '*m*' as $V_C >> V_A$. On the other hand, the number of gating transistors decreases linearly with '*m*', resulting in an almost linear decrease in the power consumption of the circuit. Note that the current requirement does not increase when the nanomagnets are stacked as they are connected in series with each other. Stacking increases the complexity of routing, as the nanomagnets need to be connected in series with the gating transistor.

Hence the degree of stacking (m) is primarily constrained by the physical limitations of routing. The nanomagnet stacking optimization for the sample benchmark of Fig. 2.11 is illustrated in Fig. 2.15. Note that the nanomagnets in each gating domain are connected in series with the source terminal of the corresponding power gating transistor.

2.2.1. Fine-grained logic pipelining

Logic pipelining is a common optimization used to improve throughput. In conventional CMOS circuits, logic pipelining incurs significant power overheads in the form of latches or flip-flops. In contrast, the nanomagnets in ASL may themselves be viewed as state elements and therefore the circuits can be pipelined in a fine-grained manner with minimal overhead. The gating domains identified in Section 2.2.3 can be utilized for this purpose. The key idea is that once a gating domain is evaluated on a given input, instead of power gating the domain, it can alternatively be used to evaluate

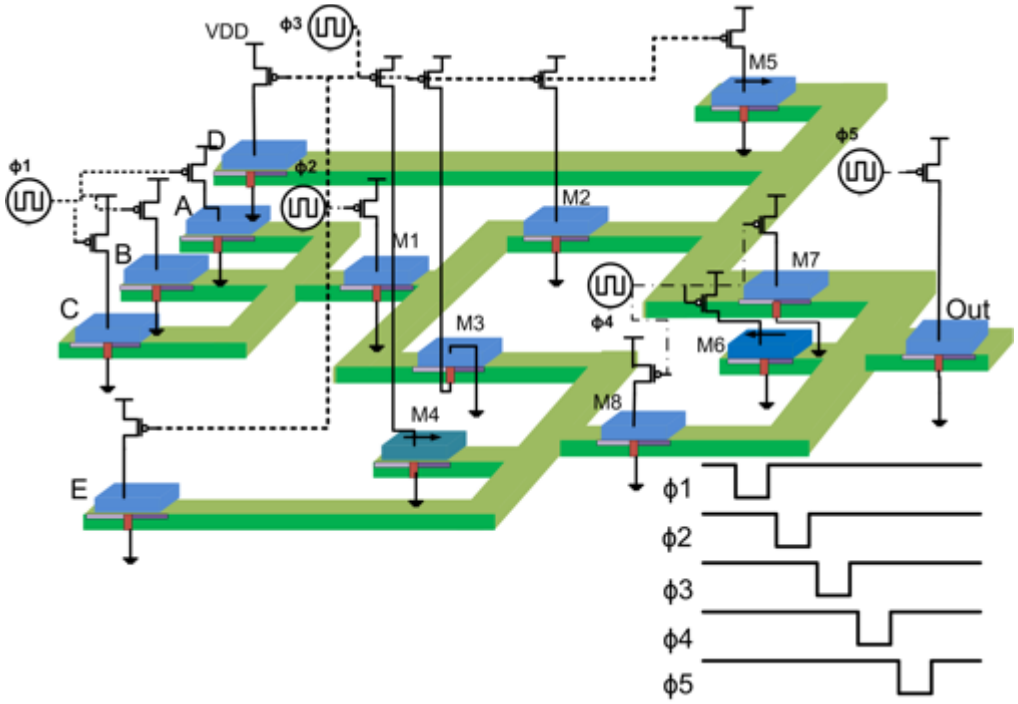


Fig. 2.14. Illustration-sample benchmark with power gating.

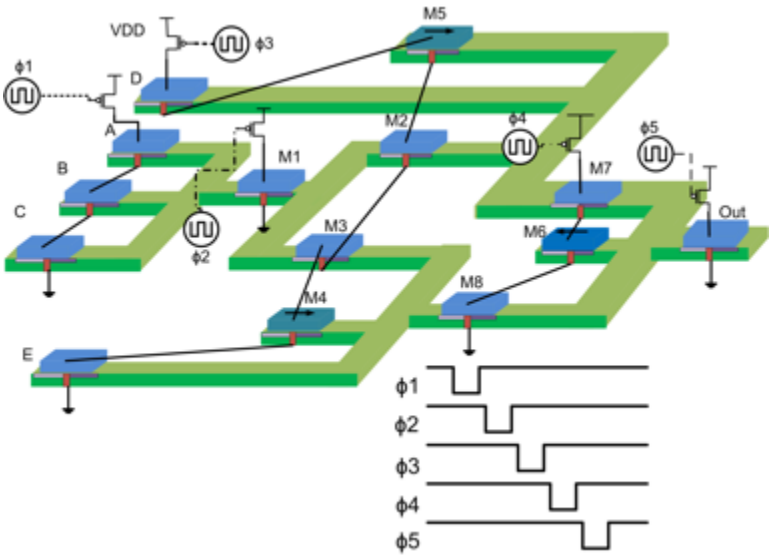


Fig. 2.15. Illustration-sample benchmark with stacking.

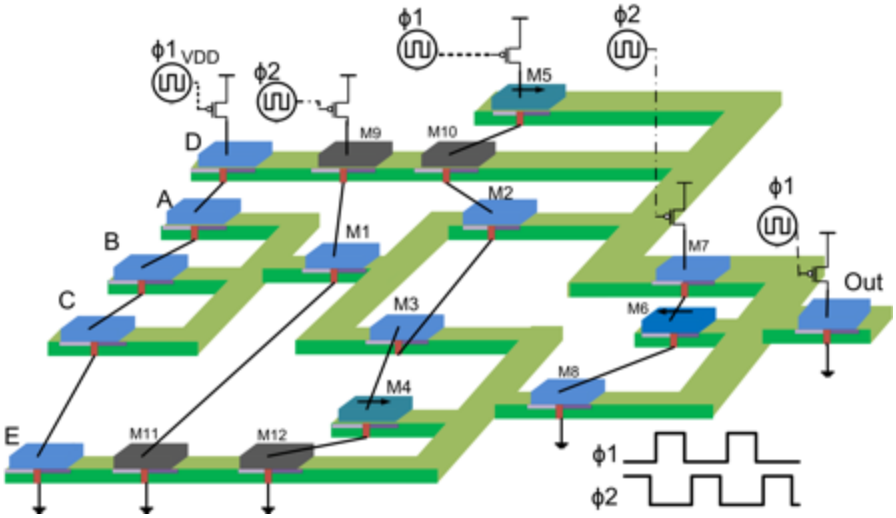


Fig. 2.16. Fine grained logic pipelining for the sample benchmark.

subsequent inputs in a pipelined manner. However, this requires several design considerations. First, successive pipeline stages cannot be operated together, as the output nanomagnets of the first stage are the input nanomagnets of the next. Hence we apply a two-phase pipelining strategy wherein only the odd stages are operated in one cycle, followed by the even stages in the next.

Another constraint for logic pipelining is that the paths in the circuit should be balanced, *i.e.,* if there exists a connection between gates that are not located on adjacent gating domains (or pipeline stages), then buffers need to be inserted in each intermediate gating domain to latch the state, as it will be overwritten in the course of pipelined

operation. The buffer insertion can be expensive depending on the number of fan-ins and fan-outs for a given gate. Hence, we utilize a modified list scheduling [35,36,37] procedure to move gates across gating domains to minimize the buffer costs. Note that, while logic pipelining significantly improves throughput of the circuit, it may not be feasible when sequential dependencies exist across different pipeline stages. The proposed methodology aggressively pipelines the circuit while ensuring such dependencies are not violated. Fig. 2.16 shows the ASL implementation of the benchmark in Fig 2.11 with fine-grained logic pipelining. Nanomagnets M9-M11 in the netlist are inserted to balance the paths in the design in order to facilitate pipelined operation.

2.3. Approximate Placement and Interconnect estimation

All Spin Logic operation is based on spin current injection through an input nanomagnet, diffusion of the spin current through the metallic channel, and updating the state of the output nanomagnet. However, the spin diffusion length of the channel material (Cu in our case) limits the flow of spin current in the channel. Therefore, if the channel length is longer than the spin diffusion length, there is a need to insert one or more buffers or repeaters (nanomagnets). Estimating the number of buffers requires a detailed place-and-route of the design, considering various physical design rules. In this work, we adopt an approximate placement methodology and estimate interconnect lengths based on this placement as described below.

Each ASL gate consists of nanomagnets communicating through Cu channels and the corresponding gating transistor(s). Note that there are two types of interconnects in ASL: (a) charge based interconnects for the gating transistors and (b) spin based interconnects (channels) for ASL logic operation, which are limited by the spin diffusion length of the channel material. The spin channels/interconnects require buffer/repeaters based on the channel length and the fanout associated with each output nanomagnet. In our analysis, we ignored any buffers required for charge based interconnects and only analyzed the buffer insertion for spin-channels to understand the design issues related to spin logic.

We estimate the spin channel lengths (and hence, the buffers) by placement of different ASL logic blocks. At first a layout of each nanomagnet was obtained and logic

gates and logic functions were placed based on the layout of a nanomagnet. Fig. 2.17 (a) shows a layout of a nanomagnet (showing the contacts) and Fig. 2.17 (b) shows a layout of an AND gate based on [38].

The placement is performed utilizing a hierarchical approach. Initially, a square layout consisting of a 3x3 grid is considered. Then, we partition the circuit into nine different partitions utilizing the *hMetis* partitioning tool [39,40].

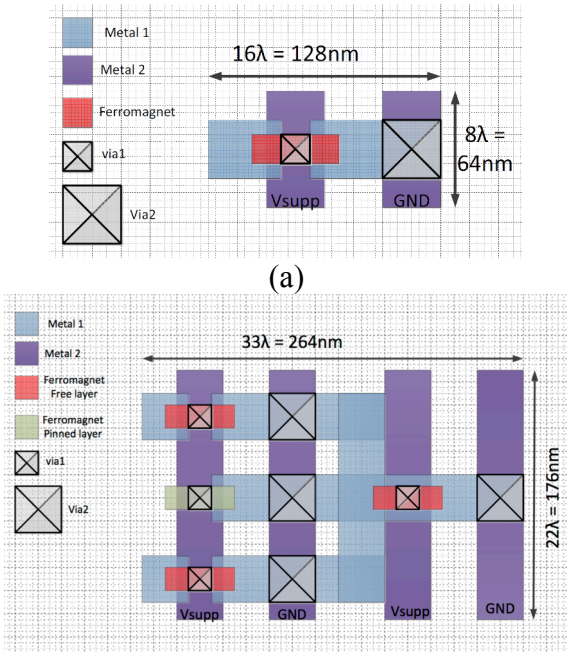


Fig. 2.17. (a) Layout of an ASL nanomagnet (b) Layout of an ASL NAND gate.

Subsequently, the inter-partition connectivities are analyzed to determine a coarse floorplan of the circuit. Each partition is placed in one of the nine sections in the grid based on the connectivities between the partitions. The number of levels of hierarchy is determined based on the size of the circuit. The same steps are repeated hierarchically: each partition is again partitioned and inter-partition connectivites are analyzed to obtain a placement of sub-blocks. Once the placement is performed, the Half Perimeter Wire Length (HPWL) [41] is used to obtain the length of each wire and if the wire exceeds the spin diffusion length, an interconnect buffer is inserted. Thus, an approximate number of required interconnect buffers isobtained. Finally, the netlist is updated using the list of interconnect buffers (and corresponding gating transistors). Note that stacking (and

pipelining for combinational circuits) is again performed to obtain a post-placement optimization of the circuit.

In summary, the design methodology described in this section enables the synthesis of ASL implementations for any given input circuit. Further, the proposed optimizations exploit several favorable characteristics of STT devices to improve energy and throughput, while trying to efficiently mask their weaknesses.

2.4. Results and Analysis

We utilized the proposed design methodology to synthesize ASL implementations for three different classes of circuits, *viz*. random combinational and sequential logic circuits, digital signal processing data-paths and a general-purpose processor (Leon SPARC3). In this section, we present and analyze the results and study the feasibility of ASL as an alternative to CMOS.

2.4.1. Experimental Methodology

Table 2.1 shows the device parameters of ASL nanomagnets utilized in the physics based simulation framework described in Section 2.3. These device parameters can be achieved with the current state-of-the-art [42,43,44]. As a representative result, an ASL inverter requires 500 μA of current for the duration of 2.5ns. The delay of ASL gates can be reduced by suitably increasing the supply current. The current is supplied to each nanomagnet using a transistor (we assumed the 16 nm technology node), with its gate terminal connected to a multi-phase gating clock of the desired operating frequency. In our evaluation, we assume the maximum number of nanomagnets that can be stacked in series in the ASL implementation to be 6. This assumption is conservative and does not considerably increase the complexity of power routing, as these nanomagnets are invariably part of at most 2-3 logic gates.

Our ASL technology library consists of 30 logic gates, with a maximum gate fan-in of 7. Each of these gates was designed and characterized using the device parameters presented in Table 2.1 We utilized Synopsys Design Compiler to obtain an initial mapping of the target design to the ASL library and developed custom tools to

Table 2.1. Device parameters of ASL nanomagnets

Parameter	Value
Damping factor (α) [44]	0.01
Magnet Volume (V)	48x16x1 nm^3
K_uV	$20K_BT$
M_S [43]	800 emu/cm^3
H_k	12.272 kOe
HighP/LowP	0.7/0.1
Channel Materials	Cu
Spin Diffusion length(λsf) [27]	500nm
Channel Resistivity	7 Ω-nm

automatically perform the various optimizations described in previous sections, *viz.* fan-out optimization, intra-cycle power gating, nanomagnet stacking, fine-grained logic pipelining and approximate placement and interconnect estimation to insert the required interconnect buffers/repeaters. We then evaluated design metrics such as area, delay and energy of the resultant ASL implementations. We note that the results include power consumed by all the components in the implementation, including the ASL gates, interconnect buffers, and power gating transistors.

Our benchmark suite comprised of three classes of circuits. The first class was random combinational and sequential logic circuits from the MCNC benchmark suite [45]. The second class of circuits was data-paths from the Digital Signal Processing (DSP) domain *viz*. 16-tap FIR filter and 8-input 1D-DCT. The final benchmark was the open source implementation of Leon SPARC3 general-purpose processor [46]. We compared the designs at two different performance targets – a moderate operating frequency of 160 MHz and a low frequency of 25 MHz. The lower frequency of 25MHz reflects the regime in which ultra-low power ICs used in sensors and implantable devices operate. On the other hand, the higher frequency was limited to 160MHz because ASL faces reliability challenges due to the large currents required to switch nanomagnets at higher speeds.

We would like to clarify the relationship between the operating frequency of the overall circuit (25/160 MHz) and the gating clock used to switch the gating transistors in each gating domain. If the circuit is not pipelined, the width of the clock pulse supplied to each gating domain should be lower than the target clock period of the circuit divided by the number of the gating domains in the implementation. This is because in each "cycle"

of operation, all the gating domains in the circuit need to be evaluated sequentially. In the cases when fine-grained logic pipelining is employed, the gating clock pulse width is less than half the target clock period, due to the two-phase pipelining scheme that requires 2 (odd and even) gating domains. Thus, the maximum operating frequency of a circuit depends on the number of gating domains in the implementation. As mentioned above, we do not target operating frequencies beyond 160 MHz, as some of the benchmarks with large numbers of gating domains cannot operate reliably due to excessive current requirements.

For the CMOS baseline, we used the 16nm PTM technology library [47] and the designs were optimized for energy using Synopsys Design Compiler. Synopsys Power Compiler and Synopsys Primetime were used for power, area and timing evaluation. It is worthy noting that the CMOS baseline is well optimized, including gate sizing and voltage scaling in the case of low frequency operation.

2.4.2. Energy and Area Comparison

In this subsection, we compare the area, performance and energy consumed by ASL circuits with CMOS implementations for different benchmark categories. We note that the results presented in this section are at the logic level and do not consider physical design information, *i.e.,* the interconnect buffers and their overheads are not included. The impact of interconnect buffers on energy and area is described in Section 2.3.

1. Random logic: Fig. 2.18 compares the energy consumption of ASL to the 16nm CMOS baselines for random combinational and sequential benchmarks operating at 160 MHz. We observe that, despite the proposed optimizations, the energy of ASL is 4-10X higher than CMOS in the case of combinational logic and two to three orders of magnitude higher for sequential logic. The primary reason behind the energy inefficiency

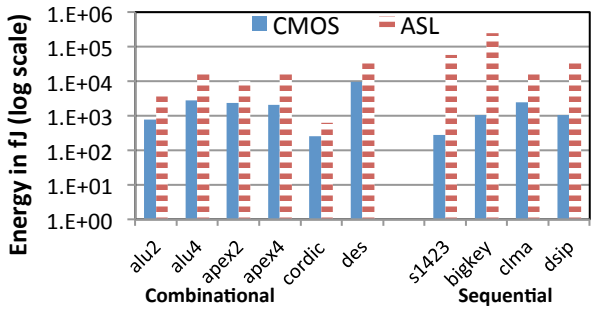


Fig. 2.18. Energy of ASL and CMOS implementations at 160 MHz.

of ASL is the large current required for switching the nanomagnets at low delays. It is worth noting that although the overall circuit operating frequency is 160 MHz, individual nanomagnets are switched at a higher frequency based on the number of gating domains in the implementation. Also, despite power-gating (Section 2.3), the power consumed by the ASL implementation is still significant. This is largely due to two factors. First, while power-gating nanomagnets reduces the period during which power is consumed, the power consumed during evaluation is still high. Second, ASL gates consume power (during their evaluation period) irrespective of whether the output nanomagnet switches from its current state. This is unlike CMOS, where the energy consumed is a strong function of the switching activity in the circuit. Since the switching activity observed in typical circuits is quite low (~0.1-0.2), a significant fraction of the overall energy consumed by ASL is attributed to power consumed even when gates' outputs do not change.

As observed in Fig. 2.18, ASL is more efficient at realizing combinational circuits since they can be pipelined in a fine-grained manner. For a given throughput, pipelining allows each stage to operate at a lower delay, requiring lower current for evaluation. On the other hand, general sequential circuits are not amenable to fine-grained pipelining due to the inherent cyclic dependencies in their structure, leading to ASL performing significantly worse for them.

Fig. 2.19 compares the energy consumption of ASL and CMOS implementations of combinational and sequential benchmarks operating at 25 MHz. In this case, we observe

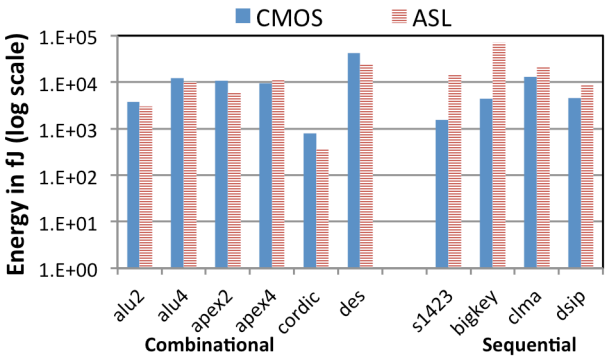


Fig. 2.19. Energy of ASL and CMOS implementations at 25 MHz.

that the combinational implementations are competitive with CMOS (10% lower energy on average), while the sequential circuits are still worse by an order of magnitude. The improved efficiency of ASL implementations at lower frequency is attributed to the decrease in the magnitude of current that needs to be passed through each nanomagnet. This is complemented by the fact that the energy benefits due to voltage scaling of CMOS implementations subside beyond a certain point, as the delay of CMOS gates begins to grow exponentially with lower supply voltage.

Finally, Fig. 2.20 presents the area of ASL and CMOS implementations. We find the ASL implementations to be, on an average, ~8X better compared to CMOS.

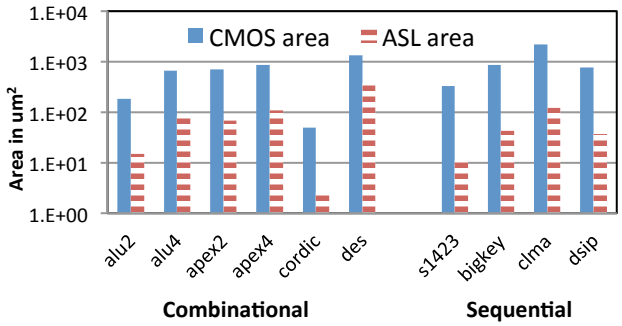


Fig. 2.20 Area of ASL and CMOS for different benchmarks.

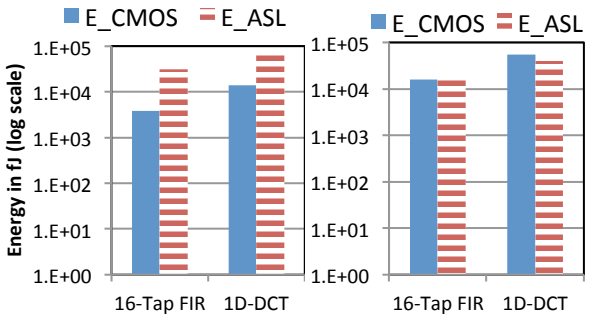


Fig. 2.21. Energy of ASL and CMOS implementations for 16-tap FIR and 1D-DCT data-paths at (a) 160 MHz and (b) 25 MHz.

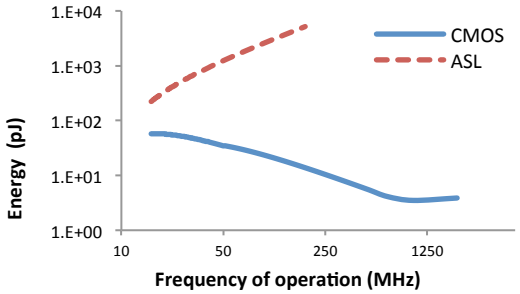


Fig. 2.22. Energy vs. delay sweep for Leon SPARC3 processor.

2. *DSP Data-paths:* Previous research efforts [28,31] have demonstrated that ASL is efficient for implementing arithmetic circuits such as adders and multipliers. Therefore, we investigated more complex DSP data-paths that are dominated by arithmetic components. Fig. 2.21 (a) and (b) show the normalized energy of CMOS and ASL implementations for the 16-tap FIR and 1D-DCT circuits at both target operating frequencies.

We observe that, while the ASL implementation is energy inefficient at 160 MHz, it provides a small energy improvement of 10% at 25 MHz for the DCT benchmark. The improvement in energy can be attributed to the ability to pipeline the DSP circuits in a fine-grained manner.

3. Leon SPARC3 General Purpose Processor: Finally, we evaluate ASL in the context of a general-purpose processor. In this case, instead of comparing CMOS and ASL implementations at just two performance targets, we perform a complete sweep of the entire design space between 2 GHz to 15 MHz and present the results in Fig. 2.22.

First, in the case of CMOS, as the frequency is decreased, the energy of the implementation also decreases due to the corresponding scaling of the supply voltage. However, beyond a certain point, the exponential growth in delay outweighs the power benefits due to the reduction in voltage, leading to a net increase in energy. In the case of ASL, it is not possible to reliably operate beyond a frequency of 160 MHz due to the large switching current requirement. We see that at 160MHz, the energy consumption is 3 orders of magnitude higher than CMOS. However, as the frequency is decreased, the overall energy of the ASL implementations proportionately decreases as smaller currents are used for ASL evaluation. At the lowest frequency of 15 MHz, the optimized ASL is still ~5X worse than CMOS. The Leon SPARC3 is an in-order pipelined processor and, while ASL can be used to efficiently implement the flip-flops in the circuit, it is not amenable to more fine-grained pipelining as feedback paths between the different pipeline stages cause complex structural dependencies in the logic. The area of the ASL implementation is 3X lower than that of CMOS.

2.4.3. Impact of Different Optimization Steps

In this subsection, we study the impact of the different optimization steps *viz.* intra-cycle power gating and nanomagnet stacking, on the energy consumption of the ASL implementation using the combinational benchmark circuit ALU2.

Intra-cycle power gating minimizes the static power consumed by the nanomagnets by switching them ON only for the duration when they are required for logic evaluation. However, as outlined in Section 2.2, it involves energy overheads in terms of gating transistors and multi-phase clocks. Thus, it is critical to identify and partition the circuit into gating domains such that the nanomagnets are switched ON for the least amount of time, while incurring minimal energy overheads. Fig. 2.23. (a) plots the energy consumption for varying number of gating domains used in the implementation for the ALU2 benchmark. We observe that, when the number of gating domains is small, the large static power consumed by the nanomagnets leads to increased energy consumption. When the number of domains is increased, the energy consumption decreases due to reduction in static power. However, beyond a certain point, the energy overheads of the gating transistors become significant, outweighing the benefits of static

power reduction. Given a circuit, our methodology identifies the right number of gating domains by analyzing this trade-off.

Next, we analyze the trade-off between stacking ASL nanomagnets and the energy consumption of the circuit. As mentioned in Section 2.3, when more nanomagnets are connected in series, the energy consumption reduces proportionately due to the orders of magnitude difference in the resistances of the nanomagnets and the gating transistor. However, since all nanomagnets in the stack are evaluated together, the time duration for which the nanomagnets in the stack are powered ON (union of the time duration of all stacked nanomagnets) may increase if the inputs to the nanomagnets arrive at different times. This results in an increase in the energy consumption of the circuit.

Fig. 2.23 (b) plots the energy *vs.* the degree of stacking for ALU2 benchmark. When the number of stacked nanomagnets is low, the energy decrease due to the increase in the degree of stacking is linear. However, when the number of stacked nanomagnets increases beyond a certain point, the energy improvements saturate, as the increase in the evaluation period nullifies the benefits due to stacking. The degree of stacking at which the benefits saturate may differ across circuits. For example, in the case of the Leon SPARC3 processor, a stacking degree of 30 yields 2X improvement over the stacking degree of 15 considered in Fig. 2.22. This is because the processor contains a large number of nanomagnets with similar time periods of evaluation and hence the increase in power ON duration for larger degrees of stacking is negligible. Given a maximum degree of stacking, our synthesis methodology evaluates this trade-off to automatically identify the optimal number of stacked nanomagnets that minimizes the overall energy consumption of the circuit.

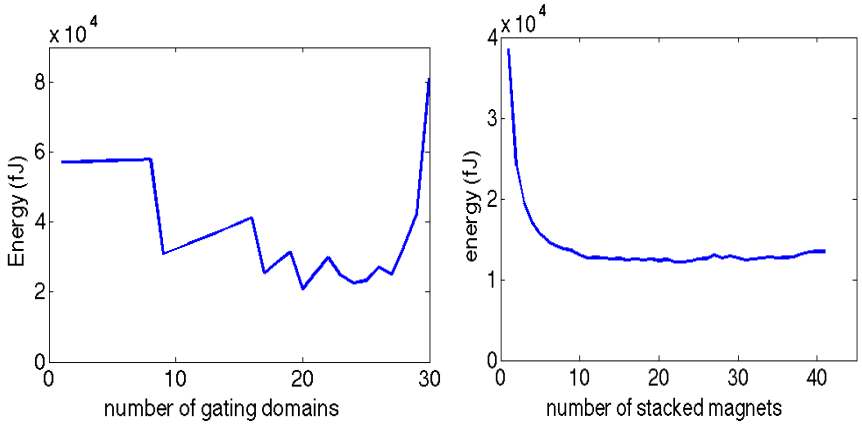
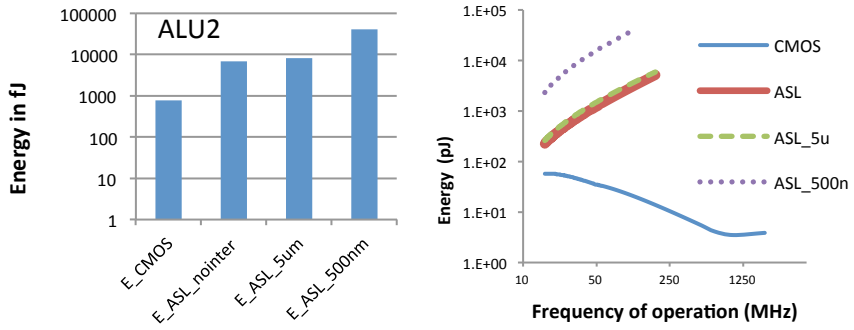


Fig. 2.23. (a) Energy *vs.* number of gating domains for ALU2 benchmark, (b) Energy consumption of ASL circuits with nanomagnet stacking for ALU2 benchmark.



(a) (b)

Fig. 2.24. Energy overhead of interconnect buffers for (a) ALU2 and (b) Leon SPARC3 benchmarks for 500 nm and 5um spin diffusion length.

2.4.4. Impact of Interconnects on ASL Energy

One of the key design constraints to ensure correct functionality of ASL circuit implementations is to account for the limited λsf. As mentioned in Section 2.3, if the interconnect length between ASL gates is larger than λsf, then appropriate number of buffers need to be inserted in order to reduce their length. In this section, we present the energy overheads of interconnect buffers obtained using the approximate place-and-route strategy described in Section 2.3.

Fig. 2.24 shows the energy of ASL implementations including the overheads of interconnect buffers for two benchmark circuits *viz.* ALU2 and Leon SPARC3. In the case when λsf is 500nm (Cu), we observe that the energy with the interconnect overheads

is ~8X compared to the ASL implementation without the overheads. However, when the diffusion length is relaxed to 5um as the case with GaAs at low temperature [48], the interconnect overheads are almost negligible. To gain further insight into the source of this overhead, Fig. 2.25 shows the wire length distribution for the ALU2 benchmark. We find that over 50% of the wires require at least one interconnect buffer. The number of interconnect buffers account for over 70% of the nanomagnets in the circuit.

Table 2.2. Current and projected device parameters of ASL

Parameter	Current	Projected [46-48]
Damping factor (α)	0.01	0.007
Magnet volume (V)	48x16x1 nm^3	15x15x1 nm^3
K_uV	$20K_BT$	$20K_BT$
M_S	800 emu/cm^3	300 emu/cm^3
H_k	12.272 kOe	12.272 kOe
HighP/LowP	0.7/0.1	1.0/0.1
Spin diffusion length (λ)	500nm	10μm
Channel resistivity	7 Ω-nm	7 Ω-nm

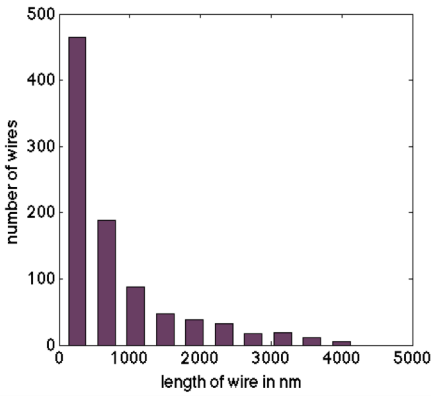


Fig. 2.25. Histogram of the wire length for ALU2 benchmark.

In addition to the direct energy contribution of the additional buffers, interconnects indirectly impact the energy consumption of ASL in several ways. For example, the long interconnects increase the logic depth of the ASL implementation as they require a chain of buffers to be inserted. Larger logic depth leads to an increased number of gating domains, which implies that for a given target frequency, the individual gating domains need to be operated at a higher frequency. This results in the entire circuit being evaluated using a larger current, and thereby causes significant energy overhead. Also, if the ASL implementation is pipelined, larger number of gating domains leads to further

additional buffers to balance all the execution paths. Thus, the λsf of the channel material can have a pronounced effect on the overall energy consumption of ASL circuits.

2.4.5. Projections of Improvements in Device Parameters on ASL Efficiency

From the above evaluations, we identify that the strength of ASL is its non-volatility, which enables it to realize sequential elements with negligible overhead. However, the large switching current required tooperate nanomagnets at high frequencies, the static power consumed in the nanomagnets during evaluation and small λsf are key bottlenecks to the energy efficiency of ASL. Clearly, ASL needs to be further optimized (through materials, device and circuit optimizations) to address these bottlenecks. In this Subsection, based on various experimental studies [46,47], we project the impact of improvements in different device parameters of that could facilitate ASL to be competitive with CMOS.

Table 2.2 shows the current and projected nanomagnet parameters. By technology scaling, the nanomagnet size can be reduced; furthermore, by utilizing lower saturation magnetization (M_S) reported in [49], the nanomagnets can be switched more efficiently. This leads to smaller evaluation current for a given target frequency. Also, improving the spin injection efficiency (High-P) from the nanomagnet to the channel material leads to larger spin torque at the output nanomagnet of the ASL gate resulting in better switching characteristics. Finally, λsf can be significantly improved by employing a different channel material such as Graphene [50]. Note that several challenges do exist in integrating/processing these materials/parameters into spin-based logic at this time. However, work has started in earnest to experiment with different channel and magnetic materials and their interfaces to improve the efficiency of ASL.

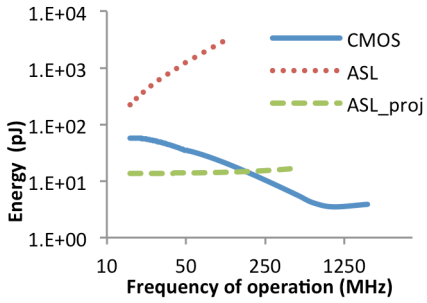


Fig. 2.26. Energy of Leon SPARC3 processor with projected parameters for ASL nanomagnets.

Based on the above projected parameters, Fig. 2.26 shows the energy consumption of the Leon SPARC3 processor at different performance targets. We find that the ASL is still not suited for high frequency operation. However, as the frequency is reduced beyond a certain point, ~200 MHz in this case, the projected ASL becomes competitive to CMOS and the improvement in energy grows at lower frequencies. At a frequency of 25 MHz, the projected ASL is ~8X better compared to CMOS. Also, a decrease in the target frequency does not improve the energy consumption as rapidly as ASL with current values of parameters. This is due to the limitation on the V_{ds} of the gating transistors, which prevents scaling the voltage beyond a certain point.

Another avenue to improve the efficiency of ASL is to exploit its lower area at higher levels of abstraction. For example, the reduction in area can be used to increase the degree of parallelism (e.g. number of cores/processing elements) in the implementation, benefiting both performance and energy.

2.5. Conclusion

All Spin Logic (ASL) is a recently proposed logic style that utilizes Spin Transfer Torque (STT) devices to realize Boolean logic circuits. While the devices possess several favorable characteristics such as non-volatility, high density and the ability to operate at low voltages, their suitability to realize arbitrary logic functions remains hitherto unexplored. Towards this end, we proposed an automatic synthesis methodology to design logic circuits using ASL. The design methodology implements three key optimizations *viz.* intra-cycle power gating, nanomagnet stacking and fine-grained logic

pipelining, all of which exploit the unique attributes of ASL to improve its energy efficiency. We utilized the proposed methodology to explore the suitability of ASL for a wide range of benchmarks, including random combinational and sequential logic, DSP data-paths and a general purpose Leon SPARC3 processor and compared ASL to CMOS ath the 16nm technology node. We identify that the delay or the switching speed of ASL nanomagnets together with the short spin diffusion length of the non-magnetic channels are key bottlenecks to their efficiency. While logic optimizations such as fine-grained pipelining are targeted to alleviate this bottleneck, they may not be possible in all cases due to the presence of sequential dependencies in the logic. In such cases, the current required to match the performance of CMOS can be quite high, and this significantly impacts the energy-efficiency of ASL. Other optimizations such as nanomagnet stacking also significantly improve the efficiency of ASL. However, the degree of stacking or the number of nanomagnets that can be stacked together is constrained by the physical limitations of routing. In summary, we conclude that ASL might show some promise for specific classes of low-power/frequency applications such as biomedical applications, internet of things *etc*. Significant improvements in the device switching times and longer spin diffusion lengths of spin channels are critical for ASL to become competitive to CMOS.

3. STT-MRAM FUNDAMENTALS

In this chapter, we will describe the STT-MRAM fundamentals including the STT-MRAM bit-cell structure and the detailed discussion of "read" and "write" operations. Furthermore, we will explain different failure mechanisms that affect the robustness of the bit-cell. These include read failures consisting of read-decision and disturbance failures, write failures and retention failures. Finally, we will briefly explain fault modeling at the array level. Furthermore, we will analyze the impact of Error Correcting Code insertion.

3.1. STT-MRAM fundamentals

3.1.1. Magnetic Tunneling Junction (MTJ)

The magnetic tunneling junction (MTJ) is a memristive device [51] and is used as the storage element in STT-MRAM. Fig. 3.1(a) shows the structure of an MTJ. As observed, the MTJ consists of two ferromagnetic layers and an oxide layer that is sandwiched between the two ferromagnetic layers. One of the ferromagnetic layers in magnetically pinned (PL) and the other one-also known as free layer (FL)- is engineered in such a way that its magnetic polarization can be either parallel (P) or anti-parallel (AP) to the PL. The MTJ resistance is low and high in P and AP orientations respectively. Furthermore, for correct operation, it is required that the energy barrier between P and AP orientations be set large enough to ensure thermal stability as well as small enough to ensure correct write operation. The MTJ is characterized mainly as a resistance [52,53,54]. R_{MTJ} depends on the MTJ geometry as well as the materials utilized for each layer. At iso-area, R_{MTJ} exponentially depends on the tunneling oxide thickness (t_{ox}). This is because the transport mechanism is tunneling. At iso-t_{ox}, the MTJ depends linearly on the cross-sectional area of the MTJ and is inversely proportional to the thickness of the ferromagnetic layers similar to an ohmic contact. Furthermore, R_{MTJ} depends on the relative polarization of the FL with respect to the PL. This is due to the difference in the

density of states around the Fermi level [55]. R_{MTJ} is lower in the P orientation in comparison with the AP orientation. Therefore, the resistance at the P orientation is called R_L ("L" stands for Low) as in the AP orientation is called R_H ("H" stands for High) The difference in the electrical resistance of the two states is called the tunneling magneto-resistance ratio (TMR= $((R_H-R_L)/R_L)$x100%).

The MTJ resistance or R_{MTJ} is important for several reasons. First of all, R_{MTJ} impacts the power consumed for read and write operations as well as the leakage power of the bit-cell; secondly, once used as a memory device, the data is encoded in terms of magnetization orientation. Therefore, the relative difference in the resistance in the P and AP states, which is included in the TMR, defines if P and AP states can be distinguished from each other. Finally, the R_{MTJ} defines the operating point of the access transistor at each orientation configuration, which from reliability point of view can impact the probability of failure of the device. This will be explained in the following subsections.

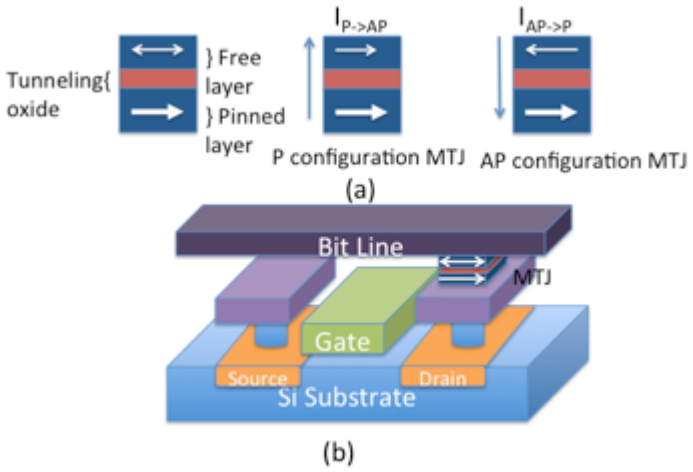


Fig. 3.1 structure of magnetic tunnel junction

3.1.2. STT-MRAM bit-cell

There are several bit-cell circuitry published in literature [56,57,58]. The bit-cell structure is shown in Fig. 3.2. As observed in Fig. 3.2, there are two different methods of connection: the standard and the reverse. In this research, we considered the standard connection between the MTJ and the access transistor. As observed, the bit-cell consists of an NMOS transistor (NFET) and an MTJ. The word line is used to turn the NFET on

or off. Once the NFET is turned on, depending on the voltage polarity between the bit-line (BL) and the source-line (SL), there is a current flow through the NFET and the MTJ. Based on the current direction and magnitude, the orientation of the magnetization reversal may occur. Magnetization reversal in STT-MRAM occurs when the current density exceeds a threshold value [59] also known as the critical current density, J_C. However, there is an asymmetry in the J_C meaning that the J_C for P to AP switching is different from that of AP to P switching. Research has shown that switching the MTJ from P to AP can be up to 50% larger than when switching from AP to P [57,60].

In order to perform read or write operations on the bit-cell, the NFET should be turned on by applying a positive voltage to the word line (WL). To perform write operation, depending on whether a "0" is being written or a "1", the bit line (BL) and source line (SL) are connected to ground or the power supply. Note that when the BL is connected to ground and the SL is connected to the power supply the NFET source and bulk are connected to the ground; however, when the BL is connected to the power supply and the SL is connected to ground, the NFET is source degenerated and the current drivability of the NFET is reduced.

In order to read from the bit-cell, the WL is connected to the power supply and a small sensing current is passed through it by applying a small voltage across the BL and SL. The current flows through a sensing amplifier which determines if the data is "0" or "1".

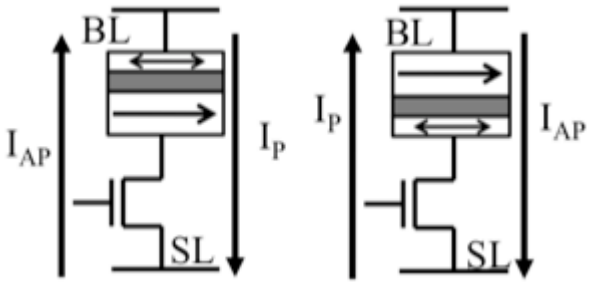


Fig. 3.2. Structure of a bit-cell: (a) Standard connection, (b) Reverse connection

3.2. STT-MRAM failures

There are four different failure mechanisms in STT-MRAMs. These failures mechanisms are write, read-disturb, read-decision and retention failures. Write failures

occur when the bit-cells have a write current density less than J_C [61]. This can occur due to weak current drivability of the NFET, which includes a small width, or a large V_T. Also, if the t_{ox} is too large, the MTJ area is too small or a combination of the aforementioned factors.

The read-disturbance failures occur when the value of the bit-cell changes during a read operation. This occurs when the current drivability of the NFET is too large, which includes a large width, or a small V_T. Also, if the t_{ox} is too small, the MTJ area is too large or a combination of the aforementioned factors [61]. Therefore, there is a conflict in the requirements for write and read-disturb operations. For example, the transistor width should be large enough to ensure a correct write and small enough to avoid a read-disturbance failure.

The read-decision failures occur when the sense amplifier senses the value of the bit-cell incorrectly. The sense amplifier compares the current that flows through the bit-cell with a reference current (I_{REF}) and if the current is lower than that, an R_H is assumed and if the current is higher than that, a R_L is assumed. Therefore, the decision failures occur when the sense amplifier outputs high for a bit-cell in P configuration (R_L) and when it senses amplifier outputs low for a bit-cell in AP configuration (R_H). For a bit-cell with an MTJ with a specific cross-sectional area and configuration, a certain t_{ox} will result in the bit-cell current to be I_{REF}. If the MTJ is in AP, a thinner t_{ox} will result in a smaller R_{MTJ} and a bit-cell current higher than I_{REF}. Therefore, the sense amplifier will incorrectly determine the bit-cell resistance to be R_L. Similarly, if the MTJ is in P configuration, a thicker t_{ox} will result in a larger R_{MTJ} and a bit-cell current lower than I_{REF}. The sense amplifier will determine the R_{MTJ} to be R_H [61].

The retention failures occur in STT-MRAM if thermal effects are large enough to change the magnet configuration of the MTJ. The stability or lifetime of the magnet is represented as t_{LIFE}, which depends on the energy barrier height of the magnet, EB, is as follows:

$$t_{LIFE} = \frac{1}{f_0} \exp\left(\frac{EB}{K_B T}\right) \tag{3.1}$$

in which f_0 is the attempt frequency and typically assumed to be 1GHz. As observed in Equation 3.1, the t_{LIFE} increases exponentially with the barrier height; thus, increasing the

barrier height enhances the retention probability of failure. However, on the other hand, the barrier height should be overcome during the write operation. Therefore, if the write energy is kept constant, increasing the barrier height would increase the write failures. Therefore, there is a trade-off between retention failures and write failures with regards to the barrier height.

3.3. STT-MRAM reliability enhancement techniques

There are different yield and reliability enhancement techniques for memories at different levels of abstraction. These yield enhancement techniques include yield enhancement techniques that target hard errors as well as the soft errors. In this research, we will briefly explain ECC codes that were utilized to enhance STT-MRAM arrays.

3.3.1. Error Correcting Codes

Utilization of error correcting codes in memory arrays is a well-know concept. However, due to their overhead in terms of power and area, there are certain codes that are more widely used for memory arrays. One of the most common ECC schemes is the hamming code. Hamming codes are "perfect codes", that is they achieve the highest possible rate for codes with their block length and minimum distance if 3 [62]. Hamming codes are used to detect and correct only a single error in the code-word. However, if an additional parity bit is added, they are capable of single error correction and double (two) error detection (SECDED). BCH codes, based on binary Galois Fields, may be used if better error correction capability is needed. Binary Galois field with degree m is represented as GF(2^m) [63]. An *(n,k,t)* binary BCH code in GF(2^m) represents a code in which $n=2^m-1$, where k is the number of bits of data that is encoded and t is the number of bits the code is capable of correcting. The probability of correction in the code word depends on the Galois field, the number of bits, and the desired correcting strength. If single error correction is desired, the number of extra bits (*t,* on top of the number of bits in the data word) is equal to that of Hamming code. If double error correction is desired, the number of extra bits increases. Furthermore, an *(n,k,t)* code can be reduced to *(n-s,k-s,t)* with the same correction capability. For example, let us assume that we have the code (31,26,1). In which the number of data bits is equal to 26. However, we want to apply this

code to a word length of 16. Therefore, it can be assumed that s=10, thus the reduced code of (21,16,1) can be used with the same capability.

3.4. Conclusion

In this chapter, we explained the STT-MRAM fundamentals including the STT-MRAM bit-cell structure and discussed “read” and “write” operations. Furthermore, we explained different failure mechanisms that affect the robustness of the bit-cell. Finally, we explained how to model the faults and the effect of Error Correcting Codes (ECC). In the next chapter, we will propose a simulation framework that optimizes the memory array at different levels of abstraction.

4. DEVICE/CIRCUIT/ARCHITECTURE FRAMEWORK FOR STT-MRAMS

In this chapter, we will propose a Device/Circuit/Architecture framework for STT-MRAMs. This chapter consists of two main sections. In the first section, we will explain the overall flow of the simulation framework as well as the methods used in each part of the framework. In the second section, we will present the results based on the framework utilization.

4.1. Cross-layer simulation framework

The developed cross-layer simulation framework that comprehends design choices at device, circuit and architecture levels of STT-MRAM design. Fig. 4.1 illustrates the flow of our cross-layer framework [64].

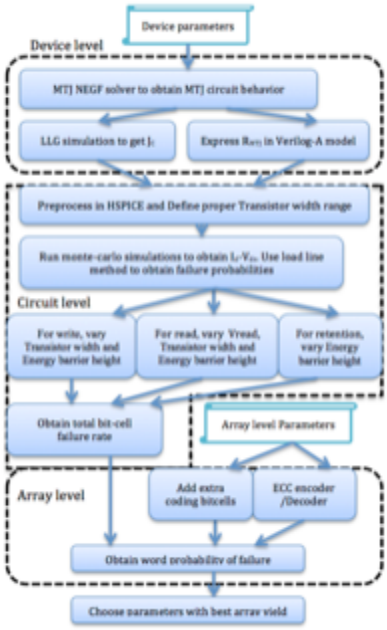


Fig. 4.1. Cross-layer simulation framework flowchart.

4.1.1. Device level modeling

The device level model was adopted from the simulator published in [65], and consists of a magnetization dynamics solver and a Non-Equilibrium Green's Function (NEGF) based electron transport solver. The characteristics of the MTJ are encapsulated in a Verilog-A model [66] that may then be used to simulate the operation of STT-MRAM bit-cells using the HSPICE circuit simulator [67]. Table 4.1 shows the device parameters assumed for this work. Furthermore, our model is calibrated to experimental data published in the literature using other bit-cell parameters taken from [61].

Table 4.1. MTJ parameters utilized in the simulation framework.

Magnetic anistropy	Perpendicular Magnetization Anisotropy
Nominal Free Layer volume	64nm x 64nm x 1nm
Oxide thickness	1nm
PMA Anisotropy Energy Barrier	$45k_BT$-$60k_BT$
Gyromagnetic Factor, γ	17.6 GHz/Oe
Saturation Magnetization, M_S	850 emu/cm^3
Damping factor, α	0.028

We have enhanced the aforementioned MTJ model to capture the thermal stability of the device, which is characterized by the lifetime, or t_{LIFE}, of the device. t_{LIFE} is also called the retention time. The probability of retention failure in a single memory bit-cell at time t is given by [68]:

$$P_{FAIL\_THERMAL} = 1 - \cosh\left(\frac{t}{t_{LIFE}}\right) e^{\left(\frac{-t}{t_{LIFE}}\right)} \quad (4.1)$$

The failure probability of n bit-cells is then given by [68]:

$$P_{FAIL\_THERMAL}(n) = 1 - (1 - P_{FAIL\_THERMAL})^n \quad (4.2)$$

4.1.2. Circuit level modeling

The circuit level model for STT-MRAM bit-cells consists of the MOSFET model and the MTJ Verilog-A model. Together, they allow the failure probability of the bit-cell to be calculated using HSPICE circuit simulations. The method in [61] is adopted in this work for calculating STT-MRAM bit-cell probability of failure. The advantage of using this approach is that the dependence of bit-cell probability of failure on the device level

parameters may be directly captured, which is different from the approach taken by prior works in the literature. Hence, our simulation framework is more suitable for evaluating the use of ECC (simultaneously considering device and circuit level designs) to improve the energy efficiency and robustness of high performance on-chip STT-MRAM [61].

4.1.3. Array level modeling

For the array level modeling, ECC was considered to enhance the reliability of the memory array. For this purpose, Hamming code with Single Error Correction and Double Error Detection (SECDED) capability and BCH code with Double Error Correction and Triple Error Detection (DECTED) capability were utilized. Furthermore, ECC overheads including extra bits required for encoding and the impact of ECC encoder and decoder were analyzed.

Table 4.2. ECC block synthesis results.

Decoder type, ECC configuration	**Hamming, (21,16,1)**	**Hamming, (38,32,1)**	**Hamming, (71,64,1)**	**Hamming, (136,128,1)**	**BCH, (26,16,2)**	**BCH, (44,32,2)**	**BCH, (78,64,2)**	**BCH, (144,128,2)**
Area (um2)	963	1370	2570	4907	1816	7924	30038	106700
Delay (ns)	0.75	0.91	1.1	1.33	0.65	0.91	1.14	3.62
Dynamic power (mW)	0.008	0.0222	0.051	0.109	0.098	0.657	2.7472	1.0919
Leakage power (mW)	0.022	0.0329	0.061	0.116	0.045	0.184	0.713	2.2346

4.2. Simulation results and discussion

The optimization methodology was utilized to evaluate a 64MB memory block with word lengths of 16, 32, 64 and 128. The bit-cells were designed using transistors from [69]. There are different parameters that affect the characteristics of the bit-cell and some of these parameters have a more pronounced impact on the reliability of the STT-MRAM array. At the bit-cell level, the barrier height was considered due to its influence on the retention failures. The second parameter was the width of the bit-cell transistor. This parameter represents the cell area as the transistor dominates the cell area.

At the array level, we utilized ECC codes. As mentioned in the previous subsection, we considered Hamming code with SECDED capability and BCH code with DECTED capability. We also synthesized the ECC encoder/decoders using the Synopsys

Design Compiler. The synthesis results and the ECC block configuration are shown in Table 4.2. Subsequently, we analyzed three types of failures: read, write and retention failures and finally, we analyzed the total probability of failure with respect to different bit-cell and array level parameters.

4.2.1. Retention Failure Analysis

As explained in Cross-layer simulation framework subsection, retention failures depend on the thermal stability of the MTJ, which is dependent on the lifetime. Fig. 4.2 plots the word retention probability of failure vs. barrier height for different word lengths. As observed in the Figure, the retention failure reduces exponentially with an increase in the barrier height. Furthermore, as it can be observed in the Figure, the $40K_BT$ barrier height has a lifetime of 7.5 years at the bit-cell level, at the word level, the probability of failure is high. Thus, the barrier height was considered to be $45K_BT$ or higher. Furthermore, it has been shown that the typical barrier height is set to $60K_BT$ [70]. Therefore, we set the higher bound to the barrier height to be $60K_BT$.

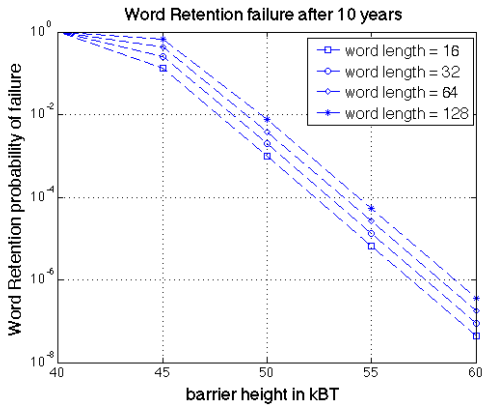


Fig. 4.2. Word retention failure vs. barrier height plot.

4.2.2. Write Failure Analysis

In order to analyze the write failures, the energy barrier height and the bit-cell area were varied at nominal Vdd. The change in the transistor width, changes the current flow through the bit-cell and thus impacts the write power consumption of the bit-cell. On the other hand, since the transistor dominates the bit-cell area, the bit-cell area is proportional to the transistor width. Fig. 4.3 shows the power consumption vs. bit-cell area. As observed, the power consumption is roughly proportional to the bit-cell area.

Fig. 4.4 shows the word probability of failure vs. bit-cell area. The effective bit-cell area is the total area of all of the bit-cells normalized to the number of unencoded data bit-cells that will be utilized during the run time application. Note that for a fixed effective bit-cell area, the probability of failure increases with an increase in the barrier height. This increase is due to the higher threshold current density required for write in a bit-cell. On the other hand, increasing the effective-bitcell area increases the write current and reduces the write probability of failure. Therefore, for write operation, it is desired to have a lower barrier height.

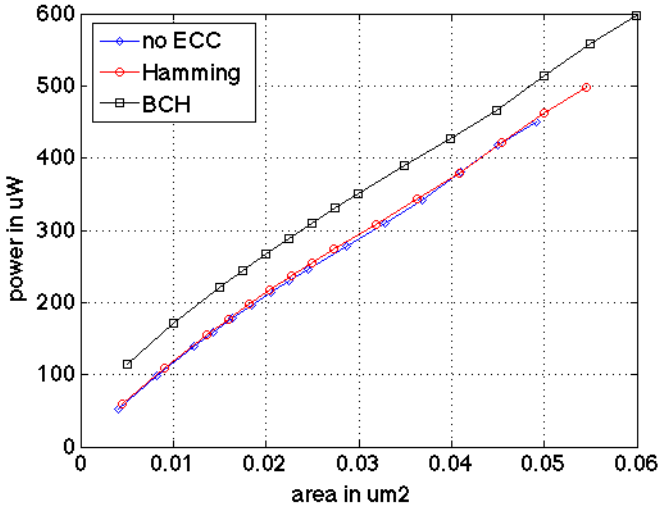


Fig. 4.3. The effective bit-cell write power vs bit-cell area.

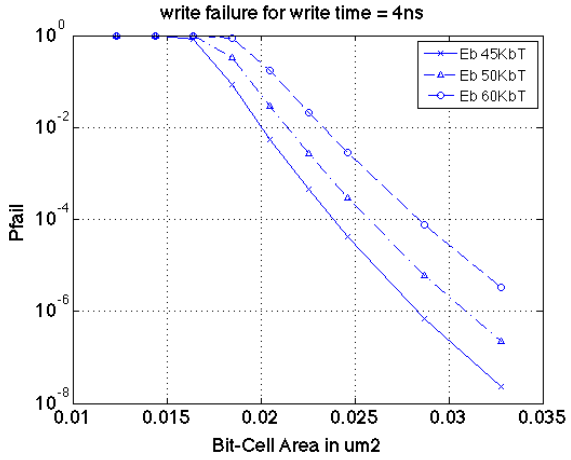


Fig. 4.4. Probability of write failure for different barrier heights.

4.2.3. Read Failure Analysis

At the next step, the read failure was analyzed against the read voltage and bit-cell area. Furthermore, the barrier height does not have an impact on the read-decision failures; however, read-disturb failures increase with a decrease in the barrier height. Thus, the worst case scenario would occur when the barrier height is minimum, which in our case is 45 K_BT. Fig. 4.5 shows the word probability of failure vs. read power and bit-cell area. As seen, the probability of failure decreases with an increase in the bit-cell area due to the enhancement in the TMR. However, bit-cells tend to get disturbed at high read voltages and high bit-cell areas.

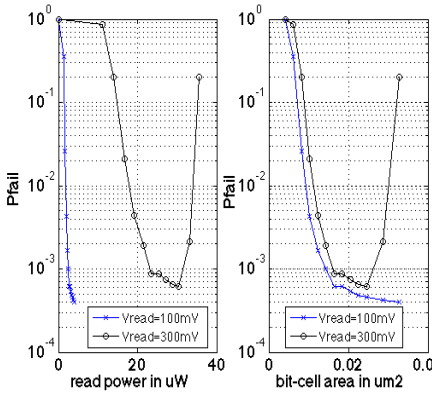


Fig. 4.5. Read probability of failure vs (a) power and (b) bit-cell area.

4.2.4. Overall Reliability Analysis

Finally, read, write and retention failures were considered jointly to truly optimize the word probability of failure. In our analysis, we have optimized the word probability of failure and not bit-cell probability of failure. Furthermore, we considered a fixed delay of 4ns for read and write operations. Besides, the total power is enumerated by giving similar weightage to read and write power consumptions (i.e., equal probability of '0' and '1' in data and equal probability of read and write occurrence).

Fig. 4.6 shows the total bit-cell probability of failure for different barrier heights and bit-cell areas for different ECC configurations. It can be observed in Fig. 4.6, that an increase in the bit-cell area would result in a decrease in the probability of failure. Furthermore, the decrease is more vigorous when the MTJ barrier height is low compared to when it is high. Additionally, the total failure probability saturates after a certain bit-

cell area. Note that the saturation value is smaller for larger barrier heights. Therefore, the probability of failure changes with respect to bit-cell area can be categorized into two different regions: The first region is the region in which the probability of failure decreases with an increase in the bit-cell area. The second region is the region in which the probability of failure remains fairly constant with an increase in the bit-cell area. The first region is the region at which the read and write failures dominate. Observe in Fig. 4.4 and 4.5 that the probability of failure decreases with an increase in the bit-cell area. On the other hand, as explained in Retention failure analysis section, the retention failure purely depends on the MTJ parameters and changing the bit-cell circuitry does not affect this type of failure.

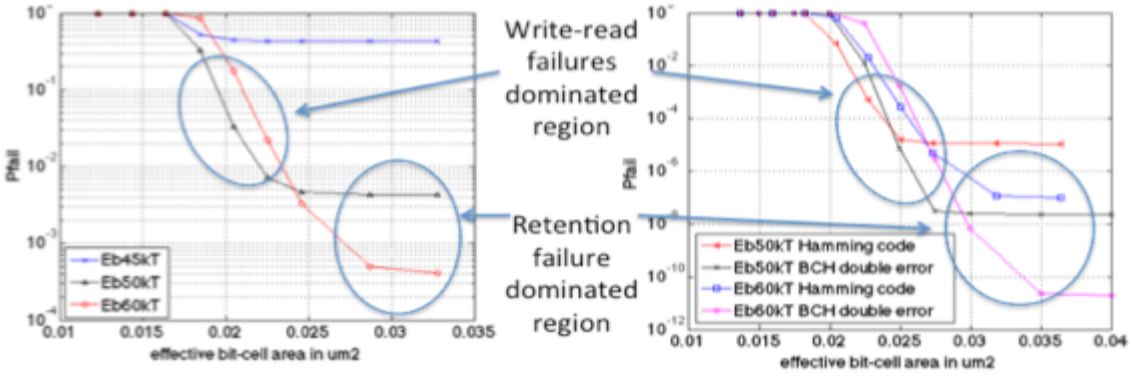


Fig. 4.6.Word probability of Error Vs. bit-cell area for: (a) no ECC (b) with ECC

At the next step, we will demonstrate the capability of this analysis. Let us assume that the design is constrained to a bit-cell area of $0.025\mu m^2$, referring to Fig. 4.6 (a), it can be observed that the best achievable probability of failure is 3e-3 utilizing MTJs with 60 K_BT. Therefore, without array level techniques, 60 K_BT would be the best choice. However, if ECC is implemented to enhance the probability of failure, Fig 5.6 (b), suggests that the best possible probability of failure at this bit-cell area is 2e-5. The intriguing fact about this fact is that the barrier height should be set to 50 K_BT and not 60 K_BT.

Let us emphasize that in a typical design in which design is performed in different levels of abstraction separately, the choice of barrier height at bit-cell level would be 60 k_BT and the best possible probability of failure after ECC application would be 2e-4,

which is an order of magnitude higher than the best possible probability of failure when the cross-layer framework is utilized. The reason to the typical suboptimal design is the limited design space at each level of abstraction. For example, at the bit-cell level, since the area is constrained and is located at the region in which retention failures dominate, the barrier height should be increased to enhance the probability of failure. Therefore, the design space is limited to the choice in the energy barrier height. However, bit-cell and array level enhancement techniques are considered jointly, a larger design space can be considered for optimal design. In the example discussed above, the cell area constraint can imposes a limit to the overall effective bit-cell area.

However, this effective bit-cell area can be assigned to the data bit-cells as well as the ECC overheads. Therefore, a larger design space is considered to achieve an optimal design of the bit-cell. On the other hand, if low power design is desired, as observed in Fig. 4.3, due to BCH coding utilization, the write power is increased by 30%-40%. On the other hand, if Hamming code is utilized instead of BCH code, the probability of failure would increase to 4e-5 and the power overhead would be reduced to 5%.

Now let us consider that the design is constrained by the probability of failure instead of the effective bit-cell area. For example, let us assume that the target probability of failure is 4e-4. From Fig. 4.6 (a) it can be observed that the minimum area that can achieve this probability of failure is $0.0325\mu m^2$ with 60 K_BT barrier height. On the other hand, Fig. 4.6 (b) shows that if ECC is utilized with the same barrier height, one can achieve effective bit-cell areas of $0.0262\mu m^2$ and $0.025\mu m^2$ for BCH and Hamming code respectively. Furthermore, if the barrier height is decreased to 50 K_BT, and by utilizing Hamming codes, the effective bit-cell area is further reduced to $0.0228\mu m^2$.

The design examples mentioned above imply that the conventional approach to STT-MRAM design, which includes optimization of the design at different levels of abstraction, leads to an over-design. Additionally, considering the design options at different levels of abstraction jointly assists the designer to explore a larger design space and to identify optimum solutions while meeting specific design constraints.

4.3. Conclusion

In this chapter, we proposed a device/circuit/architecture simulation framework and co-design methodology for STT-MRAM. Our simulation framework consists of a MTJ device level simulator, a bit-cell failure analysis model, and an array level ECC simulator. We first calibrated our simulation framework to device characteristics that were experimentally obtained and published in the literature before using it to evaluate a 64MB STT-MRAM array with 64-bit word length. The memory array was optimized by varying bit-cell parameters in conjunction with different ECC schemes. In our analysis, we found that the conventional design methodology where the bit-cell and array level designs are separately optimized yield sub-optimal designs. We found that significant improvements to the array design may be obtained using our proposed device/circuit/array co-design methodology. At iso-array area, our methodology achieved two orders of magnitude reduction in total failure rates. Moreover, our methodology can achieve 13% smaller array area at the same total array failure rate. Hence, device/circuit/array co-design is crucial for achieving dense, energy efficient and robust STT-MRAM arrays.

5. FAILURE AWARE ECC AND BIT-CELL DESIGN

In this Chapter, we propose Failure aware ECC (FaECC), which masks permanent faults while maintaining the same correction capability for soft errors without increased encoded bits. Furthermore, we investigate the impact of process variations on run-time reliability of STT-MRAMs. We provide an analysis on the impact of process variations on the life-time of the free layer and retention failures. In order to analyze the effectiveness of our methodology, we developed a cross-layer simulation framework that consists of device, circuit and array level analysis of STT-MRAM memory arrays. Our results show that using FaECC relaxes the requirements on the energy barrier height, which reduces the write energy and results in smaller access transistor size and memory array area.

5.1. Run-Time Reliability analysis

5.1.1. Thermal Stability and Retention Time

Although STT-MRAMs are referred to as non-volatile memories, their ability to retain the stored data is limited in practice. As explained in the previous Section, the retention failure probability can be expressed in terms of the time elapsed from the time data was stored in the memory and the life-time of the free layer. The life-time of the free layer can in turn be expressed as [70]:

$$t_{life} = (10^{-9})\exp\left(\frac{E_B}{K_B T}\right) \tag{5.1}$$

where E_B is the Energy Barrier height, K_B is the Boltzmann constant and T is the temperature in Kelvin. E_B depends on the geometric dimensions of the free layer. E_B for an In-plane Magnetic Anisotropy (IMA) free layer can be expressed as [71]:

$$E_B = \frac{H_k M_S V}{2} \approx \frac{4\pi M_S t(AR-1)}{wAR} M_S \frac{\pi}{4} w^2 ARt \propto t^2 w(AR-1) \tag{5.2}$$

where M_S is the saturation magnetization, H_K is the effective field anisotropy, and V is the volume of the free layer. Furthermore, w, AR and t are the width, aspect ratio and the

thickness of the free layer, respectively. As expressed in Equation 5.2, E_B depends on the physical characteristics of the free layer and is therefore sensitive to process variations. Specifically, it depends on the cross-section and the thickness of the free layer.

For Perpendicular Magnetic Anisotropy, E_B can be expressed as [70]:

$$E_B = K_{u2}V = \frac{H_k^C M_s V}{2} = \frac{H_k^C M_s \frac{\pi}{4} w^2 t}{2} \propto t\, w^2 \tag{5.3}$$

where K_{u2} is the uniaxial anisotropy, V is the volume of the free layer and H_k^C is the effective field anisotropy. As observed, the E_B depends on the volume of the free layer, which in turn depends on the thickness and cross-sectional area of the free layer.

In order to ensure reliable operation of STT-MRAM, E_B should be adjusted such that the requirements of run-time reliability are met. A typical memory reliability specification can be expressed in terms of FIT or failures in time, where 1 FIT is one failure per billion (devices × hours):

$$\frac{1}{\lambda} * (10^9) = 1\ FIT \tag{5.4}$$

where λ is the failure rate in hours and can be expressed as the equivalent of Mean Time To Failure (MTTF):

$$MTTF = \int_0^\infty t\, f_f(t)\, dt \tag{5.5}$$

where f_f is the probability density function of time to failure, if and only if this integral exists (as an improper integral).

Therefore, for an MTJ device, we have:

$$MTTF = \left| \int_0^\infty \frac{t}{t_{life}} \exp\left(-t/t_{life}\right) dt \right| = t_{life} \tag{5.6}$$

If only a single device is considered, 1 FIT translates into 0.00876% failure over 10 years, and the required E_B to meet the requirement of 1 FIT is about 50 K_BT. However, for larger memory arrays, 1 FIT should be considered for the entire memory array and not just a single MTJ device. For this purpose, let us consider a memory array with n bit-cells. Under such conditions, the probability of correctness for all of the bit-cells can be defined as:

$$F_{farray} = \prod_{k=1}^{n} F_{fk} = \prod_{k=1}^{n} \exp\left(-t/t_{life}\right) = \exp\left(-nt/t_{life}\right) \tag{5.7}$$

In which F_{farray} is the cumulative probability density function, therefore, the probability density function can be written as:

$$f_{farray} = (\frac{n}{t_{life}})\exp(-\frac{nt}{t_{life}}) \tag{5.8}$$

the *MTTF* for the entire memory array can be defined as follows:

$$MTTF = \left|\int_0^{\infty} \frac{nt}{t_{life}} \exp(-nt/t_{life})\, dt\right| = \frac{t_{life}}{n} \tag{5.9}$$

In order to obtain the required E_B, the required *MTTF* should be obtained from Equation 5.5. Next, the life-time should be defined to meet the required *MTTF* by solving Equation 5.9 for the desired array size. Once the E_B is derived, the free layer physical characteristics can be derived. In order to define the free layer characteristics, if the free layer is an IMA (PMA), Equation 5.2 (5.3) should be used. In order to analyze run-time reliability, without confining to a set of MTJ parameters, we define the normalized energy barrier height as follows:

$$E_{BN} = E_B/K_BT\ , \tag{5.10}$$

In the following Sections, we will derive the free layer physical characteristics based on the operating temperature and characteristics of the MTJ. Fig. 5.1 shows the required E_{BN} for larger memory arrays (ignoring parameter variations) for 1 FIT. As observed, the required E_{BN} increases with an increase in the memory size. Since the reliability metric of 1 FIT is kept constant, as the number of bit-cells in the memory array increases, the tolerable probability of failure for each bit-cell decreases. In order to meet this decreased probability, the E_B should be increased.

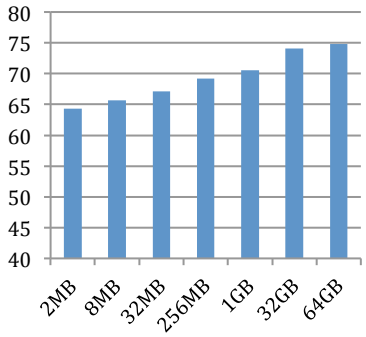


Fig. 5.1. Required E_{BN} vs. memory size for 1 FIT.

5.1.2. The effect of ECC on run-time reliability

ECC is one of the most effective methods to improve reliability of memory arrays. Among different ECC codes, Bose-Chaudhuri-Hocquenhgem (BCH) codes are commonly used in memory arrays. A BCH code changes a *k*-bit word data into an *n*-bit word data by adding (*n*-*k*) bits to the word. The choice of *n* depends on the desired correction capability of ECC. The choice of the word length at which ECC should be applied (*k*) and the extra bits (*n*-*k*) impacts the correction capability as well as the overheads incurred. The probability of correctness for an *n*-bit word with *m*-bit correction capability, with a bit error probability of P_b can be expressed as:

$$P_{word} = \sum_{i=0}^{m} \binom{n}{i} (1 - P_b)^{n-i} P_b^i \tag{5.11}$$

Furthermore, if the memory array has *s* words, every word has to be encoded using the ECC scheme selected for the memory array. Then, the probability of correctness of the entire memory array would be:

$$P_{corr} = P_{word}^s \tag{5.12}$$

In order to obtain the required E_{BN} for an array (with ECC), it is required to substitute the probability of failure of a single MTJ obtained in Equation 5.6, into Equation 5.11 as follows:

$$P_{word} = \sum_{i=0}^{m} \binom{n}{i} \left(\exp\left(-\frac{t}{t_{life}}\right)\right)^{n-i} * (1 - \exp(-t/t_{life}))^i \tag{5.13}$$

Next, the resultant P_{word} is inserted into Equation 5.12. At the next step, the probability density function is derived from the cumulative density function as follows:

$$f_{fcorr} = s\left(\frac{n}{t_{life}} \exp\left(-\frac{nt}{t_{life}}\right) + \sum_{i=1}^{m} \binom{n}{i} \exp\left(-\frac{(n-i)t}{t_{life}}\right) * \left(1 - \exp\left(-\frac{t}{t_{life}}\right)\right)^{i-1} * \frac{1}{t_{life}}\left[(n-i) - n * \exp\left(-\frac{t}{t_{life}}\right)\right]\right) * \left(\sum_{i=0}^{m} \binom{n}{i} \exp\left(-\frac{(n-i)t}{t_{life}}\right) * (1 - \exp(-t/t_{life}))^i\right)^{s-1} \tag{5.14}$$

Finally, the probability density function of time to failure is derived and inserted into Equation 5.4 to obtain the MTTF:

$$MTTF = \left|\int_0^\infty t f_{fcorr} dt\right| \tag{5.15}$$

For example, let us assume that the desired size of the memory is 4 MB and we apply ECC to every 128 bits in the array. Therefore, k=128 and s=4MB/128. For an ECC with Single Error Correction and Double Error Detection (SECDED) capability, the encoding should be performed in GF(2^8). The number of additional bits is 8 for Hamming code and a single parity bit is added to detect an additional error, resulting in a total of 9 bits. Therefore, $m=1$ and n=9+128=137. These values should be inserted in Equations 5.14 and 5.15 to obtain the *MTTF*. However, for a given MTTF, this process should be inverted to obtain the required t_{life}. Once the t_{life} is derived, it can be inserted in Equation 5.1 and 5.9 to obtain E_{BN} and E_B. Fig. 5.2 illustrates the required E_{BN} to meet the reliability level of 1 FIT for a 4 MB memory array with the word size of 128 bits. As observed, the required E_{BN} decreases with an increase in the correction capability of ECC.

Further, let us consider that ECC is employed to correct read and write failures as well as retention failures. Under such conditions, the words that happen to contain such failures (read or write) are deprived from the benefit of correcting retention failures. Moreover, if there are a large number of such words, the E_B cannot be reduced as suggested above. In the next subsection, we will analyze the impact of such conditions on the efficacy of ECC for retention failures.

5.1.3. The impact of read and write failures on the efficacy of ECC

Let us consider a scenario where ECC is used for enhancing yield as well as run-time reliability. Under such conditions, some of the words have degraded correction capability for retention failures due to the presence of other hard failures. In order to determine the impact of degraded ECC capability on run-time reliability, let us assume that the number of words with j non-retention failures is n_j. Then, the probability of correctness for the whole memory can be defined as:

$$P_{corr} = \prod_{j=0}^{m} P_{wordj}^{n_j} \tag{5.16}$$

where m is the maximum number of correctable errors and P_{wordj} can be obtained from the following equation:

$$P_{wordj} = \sum_{i=0}^{m-j} \binom{n}{i} (1-P_b)^{n-i} P_b^i \tag{5.17}$$

where n is the total number of bits in a word. For example, let us consider a 4 MB array in which ECC with SECDED capability is used. Furthermore, let us assume that ECC is applied to a word length of 128 bits,

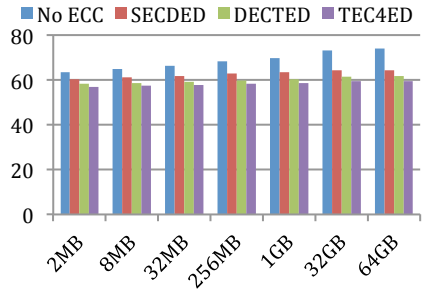


Fig. 5.2. Required E_{BN} for different memory array sizes including ECC

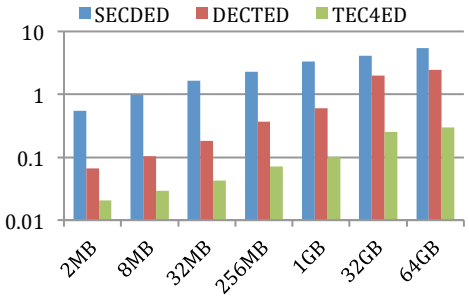


Fig. 5.3. Percentage of required increase in the E_{BN} for an array with probability of defective bit-cell of 1e-5 if ECC is utilized for yield enhancement and at run-time.

which implies that the number of words in the array is *s=4MB/16B*. Additionally, let us assume that there are b words with a single read or write failure and we have used ECC to correct these failures as well. If a retention failure also occurs in one of the b lines mentioned above, ECC with SECDED capability will be unable to correct it because there is already a read or write failure in the same word. Note, in this example, *m=1*, n_0=*s-b*, n_1=*b*. In order to calculate the MTTF, the probability of correctness obtained in Equation 5.6 should be substituted in Equation 5.17. Following these steps, the probability of correctness for the memory array can be expressed as follows:

$$P_{corr} = \exp\left(-\frac{(ns-b)t}{t_{life}}\right)\left(n + (1-n)\exp\left(-\frac{t}{t_{life}}\right)\right)^{b} \qquad (5.18)$$

At the next step, the probability density function should be derived similar to Equation 5.14.

$$f_{corr} = \exp\left(-\frac{(ns-b)t}{t_{life}}\right)\left(n + (1-n)\exp\left(-\frac{t}{t_{life}}\right)\right)^{b-1}\left[\frac{ns(n+(1-n)\exp\left(-\frac{t}{t_{life}}\right))-bn}{t_{life}}\right] \quad (5.19)$$

Finally, f_{corr} should be substituted in Equation 5.5 and the required E_{BN} and E_B can be derived.

As an example, let us consider the same 4 MB memory array with 128-bit ECC word length. Furthermore, let us assume that the probability of having a read or write failure in each bit-cell is 1e-5 and that these failures are uniformly distributed among all the bit-cells. Next, based on the uniform distribution of these failures, we obtain the mean number of words with *j* read or write failures (n_j) and insert the obtained n_j into Equation 5.18 and the probability density function should be derived similar to Equation 5.19.

Fig. 5.3 illustrates the percentage increase in the required E_{BN}. As observed, the increase in the required E_{BN} decreases with an increase in the correction capability of ECC. In order to avoid increasing E_{BN}, either an ECC scheme with higher correction capability is required or if possible, other yield enhancement techniques, such as redundant rows or columns can be introduced. However, due to poor reliability of STT-MRAM bit-cells, redundancy is not an effective method to enhance yield; it incurs high overheads [77]. Therefore, a recent trend towards ensuring reliability is to utilize ECC for yield enhancement. However, in order to maintain the high yield and run-time reliability simultaneously, there is a need to use ECC with higher correction capability, requiring higher storage area. We propose a Failure aware ECC (FaECC) scheme, which enhances the correction capability without adding large number of encoded bits to the array. In FaECC, we identify the read-decision failures, and use our proposed technique (described in the following section) to correct these failures. However, due to the stochastic nature of the write in STT-MRAMs, this method is not used to mitigate these types of failures.

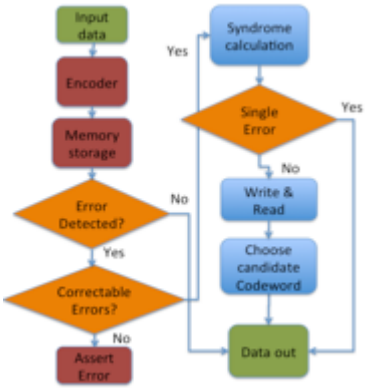


Fig. 5.4. FaECC procedure.

5.2. FaECC-based Correction

Due to their simple structure and decoding scheme, BCH codes [72,73] are commonly used in memory design. Specifically, the Hamming code, which can be viewed as a special case of BCH codes, has found widespread use in memory ECC. The number of additional encoded bits required for ECC is determined by the desired correction capability and the word length at which ECC is applied.

Furthermore, ECC can be employed to correct errors with known locations. These types of errors are called erasures [74]. The concept of potentially-erroneous bits with recognized locations is well-known and is employed in digital communications but amazingly not commonly used in memory systems [74]. Theoretically, a code with minimum Hamming distance d can correct t random errors and r erasures if $d>2*t+r$ [75]. Therefore, if we know all positions of errors, we can introduce $t=0$ and use the code for correcting erasures.

In FaECC, we use the concept of correcting erasures through ECC [74]. Fig. 5.4 illustrates the FaECC methodology. In this methodology, a SECDED code is used and erasure information are used to enable Double Error Correction (DEC) capability for stuck-at-fault errors. In this method, the encoding and decoding are performed similar to a SECDED encoding scheme and DEC decoding is enabled only if double errors are detected by the normal SECDED decoder. Once the DEC decoding is enabled, the

erasure information are retrieved and used to correct the stuck-at-fault errors. In the next Subsection, we explain the FaECC scheme in detail.

5.2.1. FaECC scheme

In Hamming code, the data bits are encoded through an encoder to obtain an encoded word (c) to be stored in the memory:

$$c \equiv a.G \tag{5.20}$$

In which *a* is the input word expressed as:

$$a = \{x_1, x_2, \dots, x_k\}, x_i \epsilon \{0,1\} \tag{5.21}$$

Note that *k* is the number of data bits. Further, *c* is expressed as:

$$c = \{y_1, y_2, \dots, y_n\}, y_i \epsilon \{0,1\} \tag{5.22}$$

in which *n* is the total number of bits to be stored including encoded bits, the expression $\equiv$ is equivalent to:

$$c = (a.G) mod\ 2 \tag{5.23}$$

The encoded word is stored in the memory. Once the code-word is read from the memory ($\tilde{c}$), it may contain one or more errors. In regular Hamming decoders, the syndrome *z* can be calculated as follows:

$$z \equiv \tilde{c}.H^T \equiv (c + e).H^T \tag{5.24}$$

where H^T is the parity check matrix and *e* is the error pattern belonging to the syndrome. The error pattern is expressed as:

$$e = \{q_1, q_2, \dots, q_n\}, q_i \epsilon \{0,1\} \tag{5.25}$$

The error pattern may contain *s* number of 1's (*s* is equal to 1 for Hamming) which corresponds to the number of errors that are corrected in the code-word:

$$e_s = \{q_1, q_2, \dots, q_n\} \mid num(q_i = 1) = s \tag{5.26}$$

To this end, every syndrome leads to exactly one error pattern with a single error. Therefore, there are *n* unique patterns possible in e_1:

$$d_1^m = \{q_1, q_2, \dots, q_n\} \iff (q_m = 1),$$

$$\forall (1 \le i, j \le n)\ d_1^i, d_1^j \in e_1, z \equiv \left(c + d_1^i\right).H^T \equiv \left(c + d_1^j\right).H^T \Rightarrow i = j \tag{5.27}$$

Meaning that each and every single error pattern would result in a unique syndrome. Furthermore, if the error pattern is all zeros, the code-word is correct, otherwise, decoding can be performed based on a syndrome table.

The same single error pattern corresponds to error patterns with 2 (duets) or 3 (triplets) errors. In order to distinguish between single error occurrence and higher number of errors, an extra parity bit is added (constructing a SECDED coding). This parity bit clarifies whether there was a single error in the code-word or two errors. If there is only one error, the decoder asserts the error based on the individual single error pattern:

$$\hat{c} = \tilde{c} + e \tag{5.28}$$

In which $\hat{c}$ is the corrected code-word. On the other hand, if two errors are detected, the error patterns can be expressed as $d_2^m \in e_2$. However, there are several double error patterns that correspond to the same syndrome:

$$\exists\ (1 \le i, j\ \le n)\ d_2^i, d_2^j \in\ e_2,$$

$$z \equiv \left(c + d_2^i\right).H^T \equiv \left(c + d_2^j\right).H^T\ \&\ i \neq j \tag{5.29}$$

Therefore, if there is no additional information, the code-word cannot be uniquely selected and the normal decoder would assert an error to the output.

On the other hand, in FaECC scheme, we consider these double-error pattern code-words and resolve which one should be considered to calculate the correct word. For this purpose, let us call the two nonzero bits in every d_2^m "active bits". Under such conditions, these error patterns are orthogonal, meaning that for every *i,j* that satisfies Equation 5.26 for the same syndrome, we have:

$$\forall\ \{i, j, i \neq j\}, z \equiv \left(c + d_2^{i,j}\right).H^T \Rightarrow\ d_2^i.d_2^j \equiv \vec{0} \tag{5.30}$$

Particularly, each of these code-words contain unique pairs of active bits; if bit *i* and bit *j* are active in code-word *x*, neither of them are active in any of the remaining code-words that satisfy Equation 5.22 for the same syndrome as *x*. In other words, each and every specific bit in the code-word is active in at most one of the possible error-patterns. In order to identify the correct candidate error-pattern, there is a need to identify the location of one of the active bits. If both of the erroneous bits are soft errors, there is

no way to find out which bits were erroneous. However, if one of the errors is a stuck-at-fault, meaning that it is possible to detect the location of the error, the correct value of both of the bits can be retrieved.

In order to find the location of one of the active bits, the erroneous code-word can be inverted and rewritten into the same line and read from it [76]. This inversion enables the decoder to detect any stuck-at-fault location and would assist in finding the correct code-word. At the next step, the code-word that was read the second time is compared to the code-word that was read at the first time and the location of the faulty bit(s) are derived. Eventually, active bits associated with the location(s) of faulty bits are fixed. Thus, the correct candidate code-word can be selected and the corrected word can be retrieved.

As explained above, the decoding scheme is capable of correcting a single error, two stuck-at-faults or a single stuck-at-fault and a single soft error. If there are two soft errors, this scheme will not be able to correct it and will assert a fault as the output. Furthermore, although we used this scheme to correct errors in STT-MRAM memory arrays, it can be used to improve the yield of any type of memory array under the aforementioned conditions.

5.3. Results and discussion

We designed a 1 MB STT-MRAM cache to evaluate the proposed ECC techniques. Tables 5.1, 5.2 and 5.3 present the parameters used for the MTJ, the decoder implementation and the characteristics of the cache respectively. In order to capture the impact of process variations on the MTJ, the volume of the free layer was considered to have a variation with a standard deviation of 2% of the nominal value. The same variation level was considered for the cross section of the MTJ. Also, in order to analyze the impact of variations on the access transistor, as explained earlier, the load line method [13] was used to obtain read and write failures.

Initially, we investigated the read operation and analyzed the impact of different parameters on read operation reliability. Fig. 5.5 (a) illustrates the probability of read failure *vs*. the access transistor width. The read decision failure probability increases slightly with an increase in the transistor width. This is due to the degradation in the bit-

cell TMR with higher transistor widths. Furthermore, the probability of failure is slightly higher for V_{read}=200mV; however, the difference is smaller than an order of magnitude. For read disturb failures, our results show that this type of failures are negligibly small for our design. Therefore, the read decision failures dominate the probability of read failure. This makes the probability of read failure virtually independent of E_B.

Next, we investigated write operation reliability. Fig. 5.5 (b) illustrates the probability of write failure *vs*. access transistor size for two different write pulse widths. The V_{dd} was considered to be 1V. As observed, the probability of write failure decreases with an increase in the width of the access transistor. This is due to the increase in the write current for higher transistor widths. Due to process variations, some of the bit-cells have higher than nominal critical current; by increasing the write current, these bit-cells are successfully written. Therefore, the probability of write failure decreases with an increase in the access transistor size. This trend is observed for both of the write pulse widths shown in Fig. 5.5 (b). However, the probability of failure is larger for the 6 ns pulse width compared to the 8 ns pulse width. This is due to the inverse relationship between the critical current of the MTJ and write pulse width – the critical current is smaller for 8 ns pulse width compared to 6 ns. Therefore, for a given nominal transistor width (which results in a given write current), the number of bit-cells with currents less than the critical current of the bit-cell is smaller for 8 ns compared to 6 ns.

Let us consider the relationship between E_B and the probability of write error. Fig. 5.6 illustrates the probability of write error with respect to access transistor width for different values of E_B for a fixed pulse width of 8 ns. As observed, for a given transistor width, the probability of error is higher for higher E_B. This relation stems from increased the critical current of the bit-cell with higher E_B.

Once the design space was explored and the bit-cells were characterized, we designed caches optimized for different design metrics. For this purpose, we considered the target yield to be 99.9% and the run-time reliability to be 1 FIT. As observed in Fig. 5.5 (a), the read probability of failure is of the order of 1e-6 and does not change substantially with change in the transistor width. Furthermore, in order to define the write probability of failure, the nominal write pulse width of 8 ns is used. As observed in Fig.

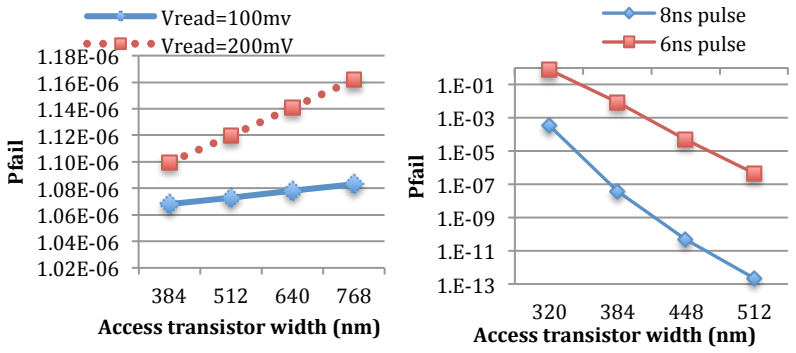
5.5 (b), the write failure increases drastically with a decrease in the access transistor width. Therefore, if the read and write probabilities of failure are considered jointly, the probability of bit-cell failure cannot be made lower than ~1e-6 by adjusting the transistor size and/or the read voltage. Therefore, it is not possible to meet the reliability target without applying ECC. This result matches the results in [15,9,77].

Table 5.1. Parameters for MTJ

Magnetization Orientation	Perpendicular
Nominal Free Layer volume	64nm x 64nm x 1nm
Oxide thickness	1nm
PMA Anisotropy Energy Barrier	50kBT-70kBT
Gyromagnetic Factor, γ	17.6 GHz/Oe
Saturation Magnetization, MS	850 emu/cm3
Damping factor, α	0.028
Temperature	300°K

Table 5.2. Decoder synthesis results.

Decoder type	SECDED (128,137,1)	FaECC (128,137,*)	DECTED (128,145,2)
Area (um2)	5433	114762	106700
Delay (ns)	1.45	10.47	3.62
Dynamic power (mW)	0.123	13.58	1.09
Leakage power (mW)	0.142	3.5	2.23



(a) (b)

Fig. 5.5. Bit-cell reliability analysis: (a) The probability of read error for two different read voltages. (b) The probability of write failure vs. access transistor width.

Next, we considered a cache with ECC. In order to have a fair comparison with respect to different ECC schemes, we considered a 128-bit ECC for SECDED and FaECC and DECTED.

Fig. 5.7 compares the energy, the area and the read/write latency for caches with different ECC configurations when the caches were optimized for minimum area. As observed, the area for a cache with FaECC is 20% and 13% less than a cache with SECDED and DECTED, respectively. Notably, the access transistor width is smaller for FaECC compared to SECDED. This reduced transistor width stems from higher coverage for read and write errors for FaECC compared to SECDED. Furthermore, since SECDED and DECTED are used for yield enhancement as well as for enhancing run-time reliability, it may not be easy to reduce E_B (run-time errors may increase). On the other hand, if FaECC is used, there exists an opportunity to optimize E_B, while still maintaining good coverage for run-time errors.

Table 5.3. Cache characteristics

Block size (bytes)	16
Associativity	1
Read/Write port(s)	1
Technology	32 nm
Cache model	Uniform Cache Architecture (UCA)

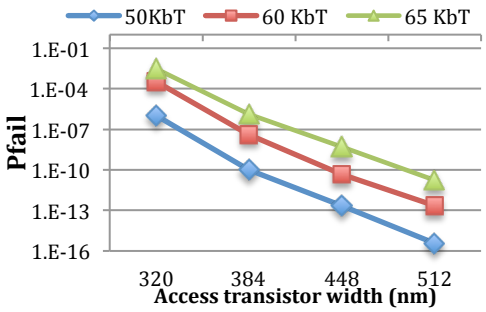


Fig. 5.6. Write error probability vs. access width for different E_B.

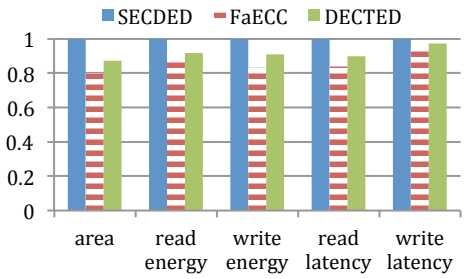


Fig. 5.7. Area, energy, and latency for caches with different ECC schemes.

Next, we optimized the cache for minimum energy consumption. In order to have a fair comparison between the three cache configurations, we considered the mean energy consumption of every read operation. Specifically, the energy associated with the error detection unit is considered for each and every read operation. However, for SECDED and DECTED, the decoder energy is considered only when an error is detected. Further, for FaECC, if a single error is detected, the decoding procedure would involve correcting a single error; thus, it would not include the additional write and read step. On the other hand, if two errors are detected, the decoding would involve extra write and read operations as well as the use of the additional decoding step. Therefore, the energy associated with each of these two conditions is added to the total energy based on the number of times each condition is applicable. Fig. 5.8 compares the read/write energy and the area of the cache after energy optimization is performed. As observed, the read energy of the cache with FaECC is 8% less than SECDED and 4% less than DECTED.

However, the write energy of FaECC is 21% and 11% smaller than that of caches with SECDED and DECTED, respectively. Also, as observed in Fig. 5.8, the area of the cache with FaECC is 20% and 11% less than the caches with SECDED and DECTED, respectively. We also optimized the cache for improved write performance. In order to have a fair comparison, we compared the mean delay of the three different ECC schemes. For this purpose, the write latency was calculated based on the latency required for a successful write operation as well as the latency for calculating the encoding bits.

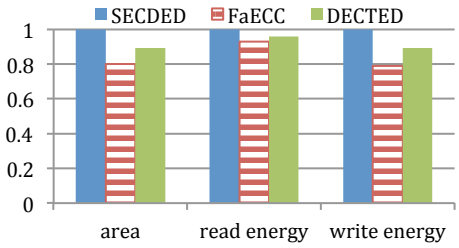


Fig. 5.8. Cache optimized for energy.

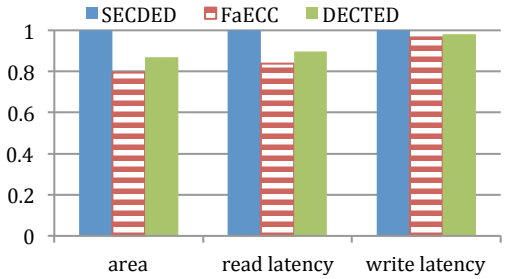


Fig. 5.9. Cache optimized for performance.

In order to obtain the mean read latency, similar to calculating the energy, we considered the weighted average of the delays of different ECC schemes based on the number of times they are invoked. For all ECC schemes, the error detection delay is included in every read operation. However, for SECDED and DECTED, the decoder delay is considered only when an error is to be corrected. For FaECC, as observed in Fig. 5.4, the data detection is performed for every data read and the correction unit is used only if there is an error.

If there were only a single error, the FaECC decoder would have the same delay as a SECDED decoder. However, the FaECC scheme differs from SECDED when two errors are detected. Additionally, this occurs only if the data value written into the hard error location is different from the data read from that location: if a bit-cell with a hard error of “0” is storing “0” (the same value) it will be read without any error. When FaECC is activated, the read latency would be dominated by a write and a read operation. This is due to parallel estimation of the candidate code-words and the additional write and read procedure. Note, that the hard error locations are required only at the end of the correction procedure. Therefore, the worst-case delay associated with FaECC is longer than that of SECDED or DECTED.

On the other hand, in STT-MRAMs, due to the long latency associated with STT switching, the write pulse width dominates the write latency. Therefore, for write performance optimization, the write pulse width should be reduced. In order to have a fair comparison, we reduced the write pulse width of all three ECC configurations to 6 ns and optimized the cache for performance. It can be observed from Fig. 5.5 (b) that if the write pulse duration is equal to 6 ns instead of 8 ns, the probability of write failure increases for a fixed access transistor width. In order to compensate for this increased probability of failure, the access transistor can be upsized to ensure complete STT switching. However, upsizing the access transistor negatively impacts the read performance. Therefore, there is a trade-off between the read performance and the write performance. Fig. 5.9 depicts the area and read/write latency for a cache with different ECC configurations. As observed, the read latency of FaECC is 16% less than that of the cache with SECDED and 11% less than that of the cache with DECTED. Furthermore, the area is 19% and 14% less than that of SECDED and DECTED, respectively.

5.4. Conclusion

In this chapter, we analyzed the impact of process variations on the run-time reliability of STT-MRAM memory arrays. Furthermore, we analyzed the efficacy of ECC in relaxing E_B requirement of the MTJ under process variations. We also analyzed the efficacy of ECC on yield enhancement and run-time reliability. Our results showed that if SECDED is used for yield enhancement besides run-time reliability, it may be difficult to have a more relaxed value of E_B (better write current). Thus, we proposed using FaECC in which permanent faults are masked while maintaining its correction capability for soft errors. In order to analyze the efficacy of FaECC, we developed a simulation framework that considers different levels of design abstraction. Using the simulation framework, we performed a case study of a 1 MB cache in 32 nm Technology node. We showed that in our proposed scheme, the area of the memory array is reduced up to 20% compared to a cache with SECDED and up to 13% compared to a cache with DECTED, at iso-reliability. Furthermore, the write energy can be reduced up to 21% and 11% compared to caches with SECDED and DECTED correction capabilities, respectively.

6. IMAGE EDGE DETECTION BASED ON SWARM INTELLIGENCE USING MEMRISTIVE NETWORKS

Recent advancements in the development of memristive devices has opened new opportunities for hardware implementation of non-Boolean computing. To this end, the suitability of memristive devices for swarm intelligence algorithms has enabled researchers to solve a maze in hardware. In this thesis, we utilize swarm intelligence of memristive networks to perform image edge detection. First, we propose a hardware-friendly algorithm for image edge detection based on ant colony. Second, we implement the image edge detection algorithm using memristive networks. Furthermore, we explain the impact of various parameters of the memristors on the efficacy of the implementation. Our results show 28% improvement in the energy compared to a low power CMOS hardware implementation based on stochastic circuits. Furthermore, our design occupies up to 5x less area.

6.1. Introduction

Bio-inspired computing has attracted a wide range of interest in the past few years for solving class of problems that are not well suited in von-Neumann architectures [78,79]. Implementation of such biological systems in standard Complementary Metal Oxide Semiconductor (CMOS) devices has turned out be energy inefficient; the inefficiencies stem from both CMOS devices and the computing platform. As an example, let us consider the simulation of cat's brain on IBM's Blue Gene supercomputer. The supercomputer consumes 8 megawatts while the cat's brain consumes only 20 watts [80]. Furthermore, the supercomputer runs two to three orders of magnitude slower than the cat's brain [80]. We believe that proper matching of devices to the algorithms can potentially lead to large improvements in energy consumption. In the quest to achieve comparable power consumption with those of biological counterparts, research has started in earnest to develop newer devices with characteristics similar to

biological elements [78,79]. Furthermore, researchers are exploring new computing models to suit bio/neuro-computing systems. Interestingly, the discovery of memristive devices has provided unprecedented similarity between electronic devices and some biological components and has enabled efficient implementation of bio-inspired algorithms [81,82]. Furthermore, researchers are exploring new computing models to suit bio/neuro-computing systems. Interestingly, the discovery of memristive devices has provided unprecedented similarity between electronic devices and some biological components and has enabled efficient implementation of bio-inspired algorithms [81,82].

Specifically, researchers have demonstrated similarities between memristive networks and *swarm intelligence* algorithms [81,82]. Swarm intelligence is the collaborative behavior of decentralized self-organized agents. These agents work simultaneously and communicate indirectly to find a solution to their problem. One of the most prominent swarm intelligence algorithms is the *ant colony optimization* method [83]. It has been shown that the ant colony algorithm is capable of efficiently finding optimal solutions to NP-complete problems such as the traveling salesman problem.

Ant colony algorithm mimics the behavior of ants to find food sources. Ants do not possess a sense of sight; however, through efficient, yet simple collaboration, they find the shortest path that leads to food sources. In order to understand the ant colony algorithm, let us consider a simple shortest path problem with two paths as illustrated in Fig. 6.1 (a). If point A is the ant nest and point B is the food source, there are two different paths to traverse from A to B. In order to find the food source, initially, ants start randomly taking different paths. To start with, roughly half of the ants take path 1 and the other half take path 2. Once they find the food source, they go back home and lay a trail of *pheromones* on their traversal path. The pheromone stays on the path for a certain amount of time and eventually evaporates. In our example, once the ants reach point B, they go back to their home, half of them go through path 1 and half go through path 2; however, since the ones going through path 1 get to their nest sooner, they lay pheromone on the path faster compared to path 2.

Note, ants favor the paths with more pheromone on them over the ones that have less pheromone. Therefore, gradually, the shortest path becomes more alluring to other ants.

On the contrary, the pheromones on the longer paths evaporate leaving them less attractive to other ants [83]. Therefore, eventually, all the ants take path 1 and the pheromone on path 1 becomes much larger than the pheromone on path 2.

Note that ants do not communicate with each other directly and on a one to one basis; however, they communicate through the pheromone that is laid on the path. This type of communication is called location-based communication. In other words, each path has a memory and remembers the traversal of the ants.

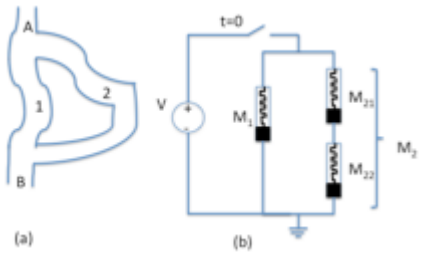


Fig. 6.1. (a) Points A (nest) and point B (food source) are connected through two paths L_1 and L_2, such that $L_2=2L_1$. (b) Memristive network model of the ant colony model in (a).

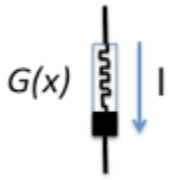


Fig. 6.2. Illustration of a memristive device.

A memristive device is a two terminal device that changes its resistance as current passes through it. For example, let us consider a simple model of a memristive device as illustrated in Fig. 6.2 [82]:

$$G(x) = G_{on} * x + G_{off} * (1 - x), \frac{dx}{dt} = K * I(t), G_{on} > G_{off}\ , 0 \leq x \leq 1 \qquad (6.1)$$

Where *G(x)* is the conductance of the memristive device, G_{on} is the minimum conductance and G_{off} is the maximum conductance. Also, *K* is the drift factor of the device and *x* is its internal state. Furthermore, *I(t)* is the current that passes through the device. If there is no current, the device keeps its current state. However, if a current passes through the device, the internal variable changes based on Equation 6.1.

For example, if $I(t)$ is a constant value, the internal variable (x) increases or decreases linearly based on the direction of the current. On the other hand, let us consider a memristive network as shown in Fig. 6.1 (b). If we consider the electrons in the circuit similar to ants and the current flow similar to ant traversal in the ant colony algorithm, they may traverse two paths in the circuit: the left path with 1 memristor and the right path with 2 memristors in series.

Furthermore, let us consider that the conductance of all the memristors is G_{off} ($x=0$) at the initial step. Additionally, let us consider that the voltage across the network is constant and equal to V_0 and the voltage is connected at time $t=0$. Therefore, initially, the current that passes through M_1 is twice the current that passes through M_2 ($I_1(0)=2*I_2(0)$). If we wish to compare this step with the initial step of the ant colony algorithm, we may consider that the ants traversing through path 2 get to B slower than the ants that traverse through path 1. Or in other words, the *density* of the ants would be smaller in path 2 compared to path 1.

Getting back to the memristive network, as explained earlier, initially, the current that passes through M_1 is twice that of M_2. On the other hand, since the rate of change in the memristive devices depends on the current that passes through them, the conductance of M_2 changes more quickly compared to M_1. Therefore, as time passes, the difference between the conductance of M_1 and M_2 becomes more pronounced. This increased change in the conductance, results in increase in the difference of the current that passes the two branches. Furthermore, the change in the current resembles the change in the number of ants that traverse path 1 due to increased pheromone after a certain period of time.

The similarity between ant colony algorithm and memristive networks was exploited in [82] to find the solution to a maze. Specifically, the authors explain that the memristive devices should be initialized with a certain resistance and propose connecting the memristive devices using MOSFETs depending on the connections in the maze; however, they fail to explain how the memristive devices should be initialized. Furthermore, they do not consider realistic models based on experimental memristive devices in literature. Besides, they do not consider real models for the MOSFET devices

and consider them as ideal switches. In this thesis, we propose using the similarities between memristive networks and ant colony algorithm for image edge detection. To this end, we make the following key contributions:

- We propose a new hardware -friendly algorithm that uses ant colony to perform image edge detection.
- We explain how ant colony algorithm for edge detection can be mapped to a network of memristive devices. For this purpose, we compare different parameters in the ant colony algorithm and explain how they can be represented as physical entities such as voltage, current and memristance of the devices.
- We simulate a memristive network based on the proposed algorithm using MOSFETs in 32nm technology and memristive devices proposed in literature [84] and analyze different design trade-offs regarding energy consumption and performance. Furthermore, we compare our results with the state of the art stochastic circuits implementation of image edge detection. Our results show 28% improvement in the energy compared to a low power CMOS implementation and occupies 5x less area.

The rest of the chapter is organized as follows. In Section 6.2, we propose a hardware-friendly algorithm for edge detection. To this end, we explain how different parameters in the algorithm impact the effectiveness of the algorithm. In Section 6.3 we propose using memristive devices for implementing the algorithm and explain the impact of various parameters on the hardware complexity of the algorithm. In Section 6.4, we describe the simulation framework for the proposed memristive implementation. In Section 6.5, we analyze the simulation results of an implementation of the algorithm using state of the art memristive devices. Finally, Section 6.6 concludes the chapter.

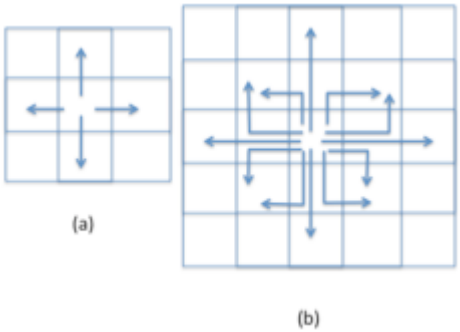


Fig. 6.6. (a) Path set illustration for L=1. (b) Path set illustration for L=2.

6.2. Implementation friendly ant colony algorithm for image edge detection

The ant colony algorithm is based on the search of multiple ants modeled as agents exploring a graph to find the optimum solution to a problem. The graph has nodes (or vertices) and edges represented as $G=(V,A)$ in which V represents the vertices and A the edges. Each edge connecting nodes i and j has two values associated with it: A heuristic, which defines the favorability of the edge (d_{ij}) and a pheromone, which mimics the pheromone in the ant colony system (τ_{ij}).

The ant colony algorithm has four main stages, namely, graph representation, initialization stage, node transition rule and pheromone updating rule.

In order to perform edge detection using ant colony algorithm, there is a need to construct a graph that represents the nature of the problem [85,86,87]. In our algorithm, we consider that the image is represented by a two dimensional graph. Furthermore, we consider each pixel as one node and assume that the pixel at ith row and jth column can be represented as $n_{i,j}$. Furthermore, we consider that there exists an edge between node $n_{i,j}$ and nodes $\{n_{i,j-1},n_{i,j+1},n_{i,j-1},n_{i-1,j},n_{i+1,j}\}$. Therefore, at each node the ant has at most four different choices to make. It also implies that the ants cannot traverse diagonally and can only traverse horizontally and vertically. However, this assumption does not affect the ability to detect diagonal edges because each diagonal edge can be considered as a horizontal step followed by a vertical step. The same assumptions are made in [86] to derive the graph representation.

The next stage is initialization. At this stage, it is required to define the heuristic associated with each edge. For each edge in the graph, the heuristics should define the favorability of the adjacent node. Since the ultimate goal of this algorithm is to detect the edges in the image, the favorability of each node is defined by the contrast of each node. However, the method of defining the contrast varies between different proposed algorithms. In this thesis, we define the heuristic associated with each node as:

$$\eta_{(i,j)} = \frac{1}{I_{Max}}\left[\begin{matrix}|I(i,j-1)-I(i,j+1)|+\\ |I(i-1,j)-I(i+1,j)|\end{matrix}\right] \tag{6.2}$$

where $I(i,j)$ is the intensity of the pixel at (i,j), I_{Max} is a normalizing factor, set to the maximum intensity variation in the whole image.

The third stage of the algorithm is simulation of ant traversal. Ant traversal is the most complex and time- consuming stage in the algorithm. Therefore, defining an effective, yet implementation-friendly algorithm is of great importance.

We suggest that ants start from each and every pixel in the image. Furthermore, the number of pixels that each ant may traverse is equal to *L.* At the next step, we define the set of all possible "paths" that an ant can traverse as the "path set". Each possible "path" from the initial point of $n_{i0,j0}$ consists of viable sequence of nodes that the ant may traverse without visiting one node more than once. Furthermore, the ant can only traverse to adjacent nodes from each and every node. Furthermore, the number of nodes in each path is equal to *L+1*. For example, if *L=1*, there are 4 paths in the path set. Each of the paths are represented with an index that shows their position in the path set. For example, the paths can be represented as: $\{path_1=\{n_{i0,j0},n_{i0,j0+1}\}$, $path_2=\{n_{i0,j0},n_{i0,j0-1}\}$, $path_3=\{n_{i0,j0},n_{i0+1,j0}\}$, $path_4=\{ n_{i0,j0},n_{i0-1,j0}\}\}$ as shown in Fig. 6.2 (a). As another example, if *L=2,* the paths can be represented as :

$\{ path_1=\{n_{i0,j0},n_{i0,j0+1},n_{i0-1,j0+1}\}$, $path_2=\{n_{i0,j0},n_{i0,j0+1},n_{i0+1,j0+1}\}$, $path_3=\{n_{i0,j0},n_{i0,j0-1},n_{i0-1,j0-1}\}$, $path_4=\{n_{i0,j0},n_{i0,j0-1},n_{i0+1,j0-1}\}$, $path_5=\{n_{i0,j0},n_{i0+1,j0},n_{i0+1,j0+1}\}$, $path_6=\{n_{i0,j0},n_{i0+1,j0},n_{i0+1,j0-1}\}$, $path_7=\{n_{i0,j0},n_{i0-1,j0},n_{i0-1,j0-1}\}$, $path_8=\{n_{i0,j0},n_{i0-1,j0},n_{i0-1,j0+1}\}$, $path_9=\{n_{i0,j0},n_{i0-1,j0},n_{i0-2,j0}\}$, $path_{10}=\{n_{i0,j0},n_{i0+1,j0},n_{i0+2,j0}\}$, $path_{11}=\{n_{i0,j0},n_{i0,j0+1},n_{i0,j0+2}\}$, $path_{12}=\{n_{i0,j0},n_{i0,j0-1},n_{i0,j0-2}\}\}$ as shown in Fig. 6.2 (b).

The next step is to describe the pheromone update rules. To this end, each edge in the graph is considered to have an initial pheromone value ($\tau_{i,j}$). Furthermore, each ant starting at each node chooses the path to traverse based on a combination of the heuristics and pheromone associated with each edge. We consider that the probability of traversing $path_m$ is equal to:

$$p_{path_m} = \frac{\prod_{(i_t,j_t)\in path_m} \tau_{(i_t,j_t)}{}^{\alpha} (1/Le_{path_m})^{\beta}}{\sum_{f=1}^{M} \prod_{(i_t,j_t)\in path_f} \tau_{(i_t,j_t)}{}^{\alpha} (1/Le_{path_f})^{\beta}},$$

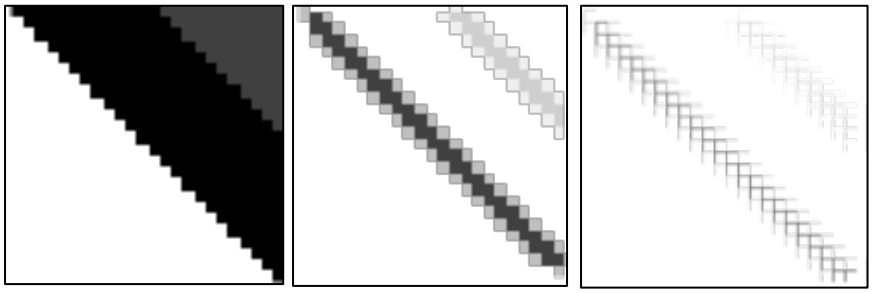
$$Le_{path_m} = \sum_{(i_t,j_t)\in path_m} \eta^{-1}_{(i_t,j_t)} \quad (6.3)$$

where $\tau_{(i,j)}$ is the pheromone leading to node *(i,j),* $\eta_{i,j}$ is the heuristics associated with node *(i,j)*, Le_{path_m} is the length of $path_m$ and α and β are two fitting parameters that define the importance of the heuristics vs. the pheromones.

Once the ant has chosen a path to traverse, the pheromone on that path is updated based on the following rule:

$$\tau_{(i,j)}(k+1) = (1-\rho)\tau_{(i,j)} + \frac{\nu Q}{Le_{path_m}} \tag{6.4}$$

where $\tau_{(i,j)}(x)$ is the pheromone at step *x*. ρ is the pheromone forget rate, Q is a fitting parameter and Le_{path_m} is the length of the chosen path. In other words, the new pheromone value depends on the old pheromone value plus a value that depends on the attractiveness of the path the ant has chosen. For example, larger (smaller) values of $\eta_{(i_t,j_t)}$ result in smaller (larger) path length and thus larger (smaller) pheromones.



(a) (b) (c)

Fig. 6.3. (a) A sample image with a two edges. (b) The bitmap image of the contrast of the image in (a). (c) The graph representation of the image in (a).

```
Initialize the edges on each pixel
For iteration=1:N
        For i=1:NumColumn
                For j=1:NumRow
                        For step=1:L
                                Select a path
                                Update pheromone
                        End //step
                End //j
        End //i
End //iteration
```

Fig. 6.4. Pseudo-code for the proposed ant colony algorithm

In order to illustrate various stages of the algorithm, let us consider the gray-scale image example image shown in Fig. 6.3 (a). In order to detect the edges of the image, initially, the contrast of each pixel is evaluated and set as the heuristic associated with each pixel as shown in Fig. 6.3 (b). Specifically, larger (smaller) values of contrast are shown in darker (lighter) gray scale color. The graph representation of the image shown in Fig. 6.3 (b) is shown in Fig. 6.3 (c). Observe in Fig. 6.3 (c) that the edges may have

several different values illustrated with different gray-scale color tones. It is noteworthy that this characteristic is different from the maze problem in which the heuristics could possibly have only two distinct values.

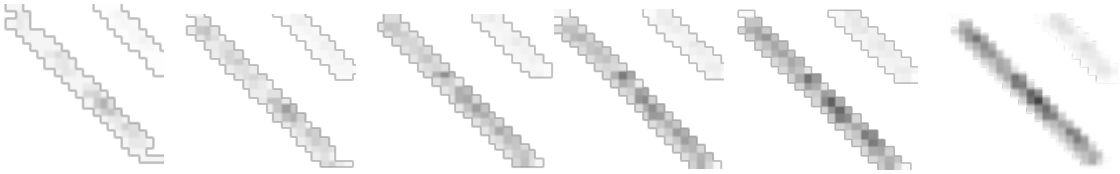
Fig. 6.4 shows the pseudo-code of the ant colony algorithm. There are several important parameters in the algorithm that have to be set correctly. First let us investigate the effect of α and β -- they define the importance of the heuristics vs. the pheromones. For now, we do not wish to emphasize the importance of one over the other. Therefore, the parameter values are set to $\alpha=\beta=1$. Furthermore, as we will explain later in Section 6.3, setting these values will ensure an exact correspondence with a memristive implementation.

Another important parameter is the pheromone forget rate (ρ). ρ defines how quickly the pheromones evaporate on each path. Setting ρ to higher values results in higher forget rates and results in slower convergence; therefore, ρ is usually set to a small value. Here we set it as $\rho=0.001$. Finally, the ant traversal length (L) should be defined. For the example problem shown in Fig. 6.3, we have set L=4. We will later elaborate more on L. A code was written in MATLAB to implement the algorithm described as a pseudo-code in Fig. 6.4. Fig. 6.5 shows the amount of pheromone deposited on each node as the algorithm progresses. As observed, the pheromones on the edges increase over time compared to pixels without any edge, and the algorithm successfully detects the edges in the image. Although the example in Fig. 6.3 is an extreme case of edge detection in a gray-scale image with three tones of color, for practical images, the same principles hold.

Now let us get back to analyzing the impact of the ant traversal length on the effectiveness of the algorithm and its complexity. As it can be inferred from Fig. 6.2, the size of the path set depends on the length of ant traversal. To this end, let us investigate the complexity of the algorithm with respect to the ant traversal length. Furthermore, let us consider that the ant starts its traversal from a node sufficiently far from the image borders. At the first step, the ant can make 4 different choices (up, down, left and right). At the next step, it can make 3 choices (because it cannot go back). At the third step, it can make the same 3 choices; however, the ant cannot traverse in a loop. Therefore, in

some cases, it can make only 1 or 2 choices. For example, it cannot make 3 consecutive right turns because it results in a traversal containing a loop. Therefore, an upper bound on the number of total choices the ant can make is $4*3^{L-1}$ for a length of L. Note, the complexity of the algorithm increases exponentially with the length of the ant traversal. Hence, from implementation point of view, reducing *L* is desirable.

Now let us investigate the impact of the ant traversal length on the quality of the detected edges. In order to analyze the effectiveness of our algorithm, test images from USC SIPI database [89] were used as sample images for the implementation.



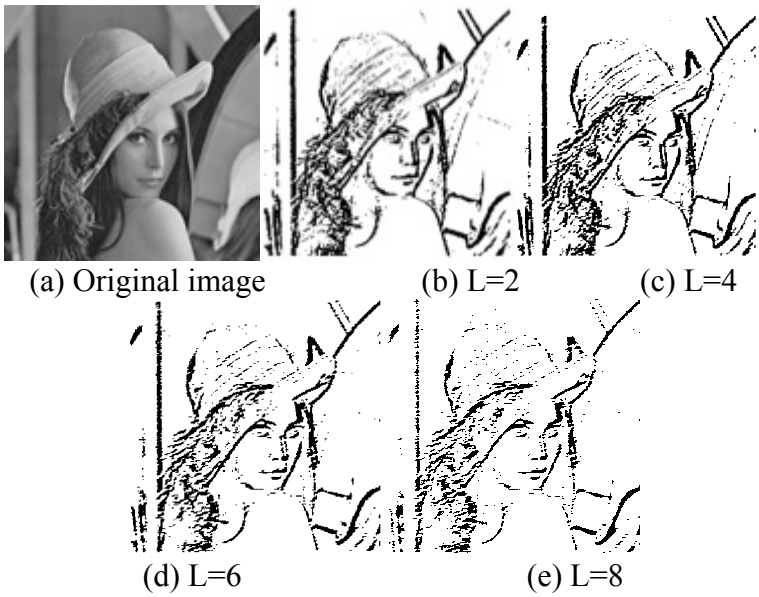
(a)Iteration=1 (b)Iteration=3 (c)Iteration=5 (d)Iteration=7 (e)Iteration=9 (f)Iteration=11

Fig. 6.5. The amount of pheromone deposited on the pixel map of Fig. 6.7 (a) as the ant colony algorithm progresses.

Fig. 6.6 shows the results of the edge detection with different *L* for the "Lena" image. As observed, for smaller values of L, the number of edges detected is higher compared to larger values of L. However, regions with high contrast are also represented as edges. These regions are observed as small black dots on the image. On the other hand, for higher values of *L*, the algorithm looks for longer edges. Therefore, very short edges are not detected in the algorithm. Thus, there is a trade-off between noise reduction and detection of short edges. On the other hand, there is a trade-off between the complexity of the implementation and the noise reduction capability. This trade-off raises the question of whether it is possible to benefit from noise reduction in longer ant traversal lengths without significant increase in the implementation complexity. One viable solution to this problem is to consider only part of the entire path set for large *L*.

For example, if *L=2*, instead of having all 12 paths, we would implement 6 of them and not the others. In other words, the ant could choose only some of the paths and not the others. To this end, we considered implementing only the horizontal and vertical

paths and not the others. Fig. 6.7 shows the edges detected using horizontal and vertical only paths for different lengths of ant traversal. As observed, for smaller values of L the algorithm performs well. However, setting the $L> 4$, has a blurring effect on the detected edges. This observation can be explained by considering the fact that at each pixel, the ant may traverse only straight towards one of the four directions around it. Furthermore, it lays pheromone on all of these edges. Setting the ant traversal length too long causes pheromone updates on pixels that are substantially far from the initial pixel of the ant; which causes a blurring effect. Therefore, this solution is only practical in ant traversal lengths that are sufficiently small.



(a) Original image (b) L=2 (c) L=4

(d) L=6 (e) L=8

Fig. 6.6. Comparison of the quality of the edges detected for the Lena picture shown in (a) for various lengths of ant traversal (L).

(a) L=2 (b) L=4 (c) L=6 (d) L=8

Fig. 6.7. Comparison of quality of the edges detected for Lena picture shown in Fig. 6.6 (a) for various lengths of ant traversal with horizontal and vertical patterns only.

6.3. Memristive implementation of the ant colony algorithm for edge detection

In this Section we propose a memristive implementation of the ant colony algorithm. To this end, we will explain how the similarities between memristive devices and the location-based communication of ants can be exploited to implement the algorithm efficiently. To this end, we first investigate a small simple edge detection problem and show how this simple problem can be mapped to a memristive implementation. At the next step, we propose a systematic approach to use the similarities for image edge detection using memristive devices.

6.3.1. Simple edge detection example

In order to show the similarities between the ant colony algorithm and image edge detection, let us focus on the progress of the algorithm using a simple image edge detection example. For this purpose, let us consider the algorithm proposed in Section 6.2 for a very small image. Fig. 6.8 (a-d) illustrate the original and a noisy image sample and their contrast images. If we wish to use the ant colony algorithm for image edge detection in the noisy image in Fig. 6.8 (c), the first step is to derive the contrast of each pixel as shown in Fig. 6.8 (d). The next step is to simulate the ant traversal. Let us consider only one ant starting from the center of the image as shown in Fig. 6.8 (e). Also, let us set the ant traversal length to $L=4$ and the pheromones on all the pixels to τ_0. Also, let us consider that there are only purely horizontal and purely vertical paths in the path set. In other words, the ant can traverse to up, down, left or right directions for 4 steps. Furthermore, let us consider that the pixels with a white color have η_0=1, the ones in grey have η_1=5, η_2=10 and η_3=15 depending on their intensity as illustrated in Fig. 6.8 (e). Under such conditions, the length associated with each path can be written as:

$$Le_{up} = Le_{left} = \frac{4}{\eta_0} = 4, Le_{right} = \frac{1}{\eta_0} + \frac{3}{\eta_2} = 1.3,$$
$$Le_{down} = \frac{2}{\eta_0} + \frac{1}{\eta_1} + \frac{1}{\eta_3} = 2.266 \quad (6.5)$$

Observe in Equation 6.5 that the length of ant traversal to up and left directions is substantially larger than the length of the ant traversal to right and down directions. In

order to simplify the example, let us consider the length of ant traversal to up and left directions to be infinity and the probability of traversal to these two directions to be zero. On the other hand, the probability of traversing to the right and down directions can be written as:

$$p_{right} = \frac{(\tau_0^4)^\alpha (1/Le_{right})^\beta}{(\tau_0^4)^\alpha (1/Le_{right})^\beta + (\tau_0^4)^\alpha (1/Le_{down})^\beta},$$

$$p_{down} = \frac{(\tau_0^4)^\alpha (1/Le_{down})^\beta}{(\tau_0^4)^\alpha (1/Le_{right})^\beta + (\tau_0^4)^\alpha (1/Le_{down})^\beta}, \quad (6.6)$$

Where τ_0 is the initial pheromone on each node, Le_d is the length of ant traversal to direction d and α and β are two fitting parameters. Now let us consider the path to the right as *path 1* and the path downward as *path 2*. Additionally, the pheromones of the paths can be represented as the product of the pheromones on all of the constituent nodes in each path. Therefore, at the first time step, we have:

$$\tau_1 = \tau_2 = \tau_0^4 \quad (6.7)$$

Therefore, rewriting Equation 6.3 considering Equation 6.6, 6.7 results in:

$$p_{1(2)} = \frac{\tau_{1(2)}^\alpha (1/Le_{1(2)})^\beta}{\tau_1^\alpha (1/Le_1)^\beta + \tau_2^\alpha (1/Le_2)^\beta} \quad (6.8)$$

in which $\tau_{1(2)}$ is pheromones laid on each path and Le_i is the length of ant traversal in the ith path. Besides, the pheromone dynamics on the first (second) path is:

$$\tau_{1(2)}(k+1) = (1-\rho)\tau_{1(2)}(k) + \nu Q/Le_{1(2)} \quad (6.9)$$

Where ρ is the pheromone forget rate and ν and Q are two fitting parameters. Now let us assume that the number of ants entering the image has a constant rate, γ. Then, the amount of ants added within a time interval dt is equal to γdt. Therefore, Equation 6.9 can be rewritten as [4,5]:

$$\frac{d\tau_{1(2)}}{dt} = -\gamma\rho\tau_{1(2)} + p_{1(2)}\gamma\nu Q/Le_{1(2)} = \quad (6.10)$$

$$-\gamma\rho\tau_{1(2)} + \frac{\gamma\nu Q}{Le}_{1(2)} \frac{\tau_{1(2)}^\alpha (1/Le_{1(2)})^\beta}{\tau_1^\alpha (1/Le_1)^\beta + \tau_2^\alpha (1/Le_2)^\beta}$$

In order to implement the ant colony algorithm using memristive devices, let us consider that a memristive device is used to represent each path in the path set as shown

in Fig. 6.8 (e). Also, let us consider that the conductance of each memristive device can be represented as:

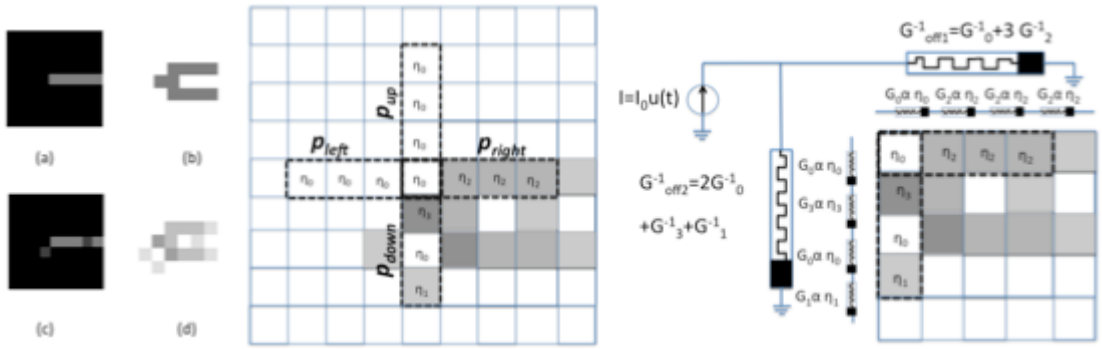
$$G_d(x) = G_{on\_d} * x + G_{off\_d} * (1 - x), G_{on\_d} > G_{off\_d} \quad (6.11)$$

where $G_d(x)$ is the conductance of the memristive device, G_{on_d} is the conductance of the memristive device in the ON state and G_{off_d} is the conductance of the memristive device in the OFF state and x is the internal variable of the memristive device. Besides, let us consider that the equation for the internal variable should contain a drift term (K) that formulates the dependence of the internal state on the current passing through it as well as a relaxation term (ξ):

$$\frac{dx}{dt} = KI(t) - \xi x \quad (6.12)$$

in which K is the drift constant and ξ is the relaxation term.

Also, the initial conductance of all of the memristors is equal to G_{off_d} where d can take four values: up, down, left and right. Besides, the value of G_{off_d} is inversely proportional to the length of each path:



(e) (f)

Fig. 6.8. Illustration of memristive implementation of ant colony algorithm for a small image. (a,b) original image and its edges. (c,d) noisy image and its edges. (e) probability of ant traversal for all of the paths in the path set. (f) Memristive implementation of (e).

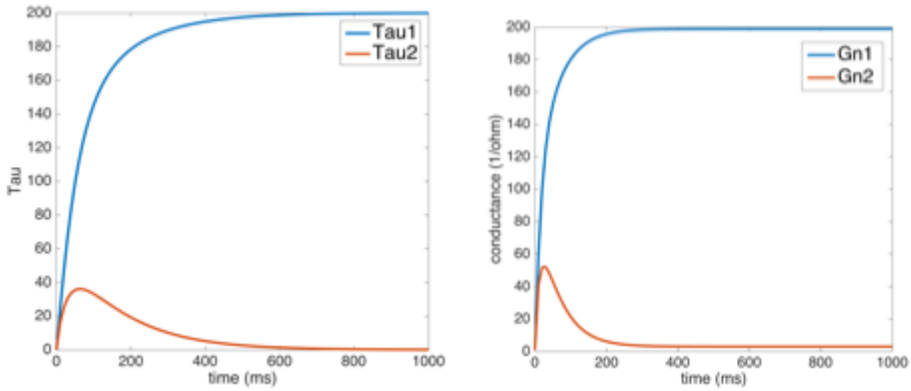


Fig. 6.9. Comparison of pheromone values and the normalized conductance in Equation 6.6 and 15 respectively. (a) parameters used are $\gamma=20$, $\rho=1$, $L_1=1.3$, $L_2=2.266$, $\tau_1(0)=\tau_2(0)=0.01$, the parameters are adjusted for illustration purposes. (b) $I_0=1$, $\xi=50$, $G_{off1}=2.266$, $G_{off2}=1.3$, $G_{on1}=2200$, $G_{on2}=1300$, $Gn_1(0)= Gn_2(0)=1$ the parameters are adjusted for illustration purposes only.

$$G_{off\_up} = G_{off\_left} = 1/4, G_{off\_right} = 1/1.3\,, G_{off\_down} = 1/2.266 \tag{6.13}$$

In order to analyze the similarities between the memristive network and the ant colony algorithm, there is a need to analyze the dynamics of the circuit shown in Fig. 6.8 (f) with that of the ant colony algorithm. Observe in Equation 6.13 that G_{off_up} and G_{off_left} are very small. In order to simplify the illustration and keep the correspondence with the ant colony algorithm example, let us consider that $G_{off\_up}=G_{off\_left}=0$. Therefore, they are considered to be open circuit. Therefore, the circuit implementation of the right and down direction paths in Fig. 6.8 (e) are illustrated in Fig. 6.8 (f). Furthermore, let us name G_{right} as G_1 and G_{down} as G_2. The current passing through each branch illustrated in Fig. 6.8 (f) can be written as:

$$I_{1(2)} = I_0 \frac{G_{1(2)}}{G_1+G_2} \tag{6.14}$$

On the other hand, in order to analyze the dynamics of the network, we assume $G_{off1}= G_{off\_right}$ and $G_{off2}= G_{off\_down}$ as initial conditions. Eventually, rewriting Equation 6.12 considering Equation 6.14, the normalized conductance of each branch and considering $Gn_i(t) = G_i/G_{offi}$ can be written as:

$$\frac{dGn_{1(2)}}{dt} = -\xi\left(Gn_{1(2)} - 1\right)$$

$$+KI_0\left(\frac{G_{on1(2)}}{G_{off1(2)}}-1\right)\frac{Gn_{1(2)}(t)G_{off1(2)}}{Gn_1(t)G_{off1}+Gn_2(t)G_{off2}} \quad (6.15)$$

Comparing Equation 6.15 and 6.6, it can be concluded that the ant colony algorithm is implemented in Equation 6.15 with parameters α=1 and β=1. Furthermore, the initial off ($G_{off1(2)}$) state can be interpreted as the heuristics associated with each path ($\eta_{1(2)}$). Besides, the normalized conductance $Gn_i(t) = G_i/G_{offi}$, which is proportional to the internal state variable, plays the role of the pheromone strength $\tau_{1(2)}$. Fig. 6.9 (a) and (b) illustrate the value of pheromones in Equation 6.6 and the value of the normalized conductance in Equation 6.15 respectively. Observe in Fig. 6.9 that the pheromones and the normalized conductance show a similar behavior and settle to a final state similarly. Furthermore, observe in Fig. 6.9 (a) that although there is a high contrast pixel in the downward direction, the pheromones on this path settle to a small state showing that this path is not an edge. Therefore, the algorithm successfully distinguishes between a real edge and that caused by a noisy pixel. Similar assumptions can be made for the memristive implementation in Fig. 6.9 (b).

However, there are differences between these two systems such as the final relaxation state. In the ant colony system, the final state is zero for undesired paths; however, in the memristive implementation, the final relaxation state is a small positive non-zero number. Despite, all these differences, it has been shown that these two systems come to the same solution [81,82].

On the other hand, it has been shown that similar results are achievable if the current source would be replaced by a voltage source. For example, if a voltage is applied to a memristive network representing a maze, the final solution can be obtained in a similar fashion [81].

Additionally, the aforementioned proof is only viable for one ant traversing from a specific pixel; nevertheless, the traversal of several ants starting from various locations in different orders is not analyzed. Finally, the proof provided in Equation 6.11-6.15 can be rewritten for voltage-based memristive devices and similar results can be obtained using these devices. Specifically, using source conversion, all of the series connections should be changed to parallel ones and the current source should be transformed to voltage

source. In the next subsections, we will propose a systematic method for image edge detection using voltage based memristive devices based on ant colony algorithm.

6.3.2. Graph mapping to memristive network

Every ant colony problem is represented as a graph explaining the nature of the problem. In order to solve the problem using memristive devices, the graph should be mapped to a memristive network. To this end, we consider that each and every pixel is represented as a memristive device. Furthermore, we assume that the memristive device at *i*th row and *j*th column can be represented as $Me_{i,j}$. At the next step, we consider that $Me_{i,j}$ may be connected to $\{Me_{i,,j-1}, Me_{i,j+1}, Me_{i,j-1}, Me_{i-1,j}, Me_{i+1,j}\}$. Fig. 6.10 (a) illustrates the required circuitry for each and every pixel. The circuitry consists of the memristive element ($Me_{i,j}$), initialization circuitry, which is used to initialize the memristive device, ant traversal simulation circuitry which is used to simulate the ant traversal and the read circuitry, which is used to read out the value of the memristive device once the stopping criterion is reached. In the following Subsections, we will explain each circuitry with respect to its functionality.

6.3.3. Initialization of the memristive network

The main goal of the initialization step is to program the memristive devices based on the definition of the problem. As explained earlier in Equation 6.13, the initial conductance of the device defines the favorability of each pixel with respect to the edge detection problem. Therefore, the initial value of the conductance is proportional to the contrast of each pixel as explained in Equation 6.2:

$$G_{ini(i,j)} \propto \eta_{(i,j)}^{-1} \tag{6.16}$$

where $G_{ini(i,j)}$ is the initial conductance of the memristor.

Activating the initialization circuitry for each pixel performs the initialization step. For this purpose, M_{ini} is used to connect the memristive device to the initialization circuitry. Furthermore, the source-line (*SL*) is pulled up to V_{dd}. On the other hand, the amount of time M_{ini} is ON defines the value of G_{ini}. Specifically, longer (smaller) ON times result in larger (smaller) changes in the internal variable. Therefore, the ON time should be adjusted according to the value of each pixel.

6.3.4. Ant traversal

At the next step, ant traversal is simulated. Ant traversal includes mapping the traversal rules and pheromone update rules to the connections and sequence of operation in the memristive network.

In order to simulate node transition, we consider connecting proper memristive devices to other memristive devices and to the power source(s). Specifically, we assume that the length of the traversal for each ant is L. As explained in Section 6.2, the ant may traverse through different *paths* in the *path set*. In order to simulate ant traversal through each *path,* memristors are connected in one of the *paths*.

In order to simulate pheromone update; it is considered that each ant traverses a specific path at a time. As explained in Equation 6.15, the change in the value of the internal variable in the memristive device is interpreted as the change in the pheromone value. Therefore, the memristive devices at each path are connected to a current source.

The current source causes a change in the internal variable of the memristive element. Ant traversal circuitry is used to realize the connections. Observe in Fig. 6.10 (a) that the ant traversal simulation circuitry consists of three transistors. Notably, M_L and M_U are used to connect each memristive device to the adjacent memristive devices horizontally and vertically. Furthermore, M_{DD} is used to connect the devices to the current source I_{update}. This current source is used to change the internal state of the memristive device.

Fig. 6.10 (c) shows the connections required for the pattern shown in Fig. 6.10 (b). Specifically, the wires shown in black are conducting and the ones in grey are disconnected. Observe in Fig. 6.10 that the source-line (*SL*) is grounded during the ant traversal simulation.

The ant traversal is simulated for a single path at a time. Furthermore, it is considered that ants start traversing the image in non-overlapping patterns. The reason for this consideration is that the ant traversal can be simulated in a massively parallel fashion throughout the image. For example, let us consider that the ant traversal length is equal to three pixels and we wish to simulate a horizontal pattern of three pixels. The ant traversal for all of the ants in the image is performed in three phases. In the first phase, we

consider that ants start their traversal from pixels at *{(i,9j+1),(i,9j+4),(i,9j+7)}* columns only and they traverse to the right. Therefore, memristive devices are connected in three *{{(i,9j+1), (i,9j+2), (i,9j+3)}, {(i,9j+4), (i,9j+5), (i,9j+6)}, {(i,9j+7), (i,9j+8), (i,9(j+1))}}* where *i* is the row of each pixel and *9j+x* is the column of the pixel. In the second phase, it is considered that the ants start from the pixels at *{(i,9j+2),(i,9j+5),(i,9j+8)}* and traverse to the right. Therefore, they are connected in groups *{{(i,9j+2), (i,9j+3), (i,9j+4)}, {(i,9j+5), (i,9j+6), (i,9j+7)}, {(i,9j+8), (i,9(j+1)), (i,9(j+1)+1)}}*. In the third phase, it is considered that the ants start from the pixels at *{(i,9j+3),(i,9j+6),(i,9(j+1))}* and traverse to the right. Therefore, they are connected in groups *{{(i,9j+3), (i,9j+4), (i,9j+5)}, {(i,9j+6), (i,9j+7), (i,9j+8)}, {(i,9(j+1)), (i,9(j+1)+1), (i,9(j+1)+2)}}*. Fig. 6.11 shows the ant traversal simulation for a purely horizontal pattern of three pixels for the two different phases.

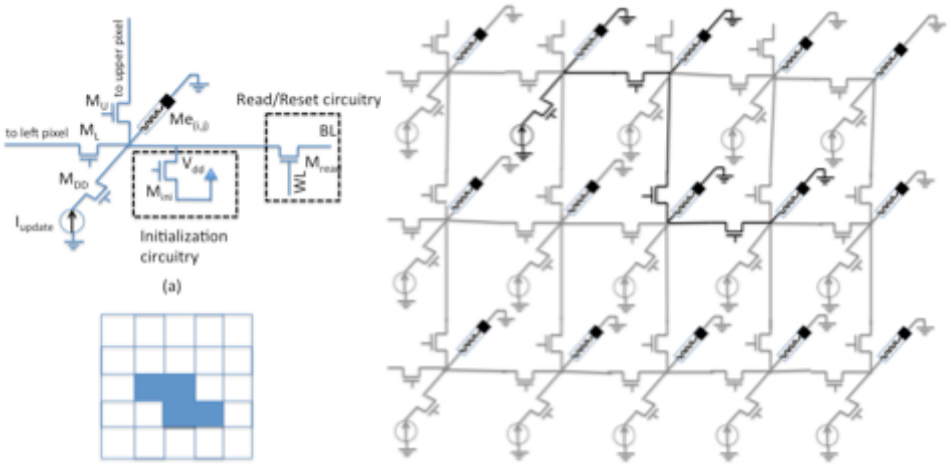


Fig. 6.10.Illustration of memristive implementation. (a) Implementation of each pixel for voltage based memristive devices. (b) A sample pattern of ant traversal. (c) Illustration of connections of adjacent pixels for voltage based memristive devices.

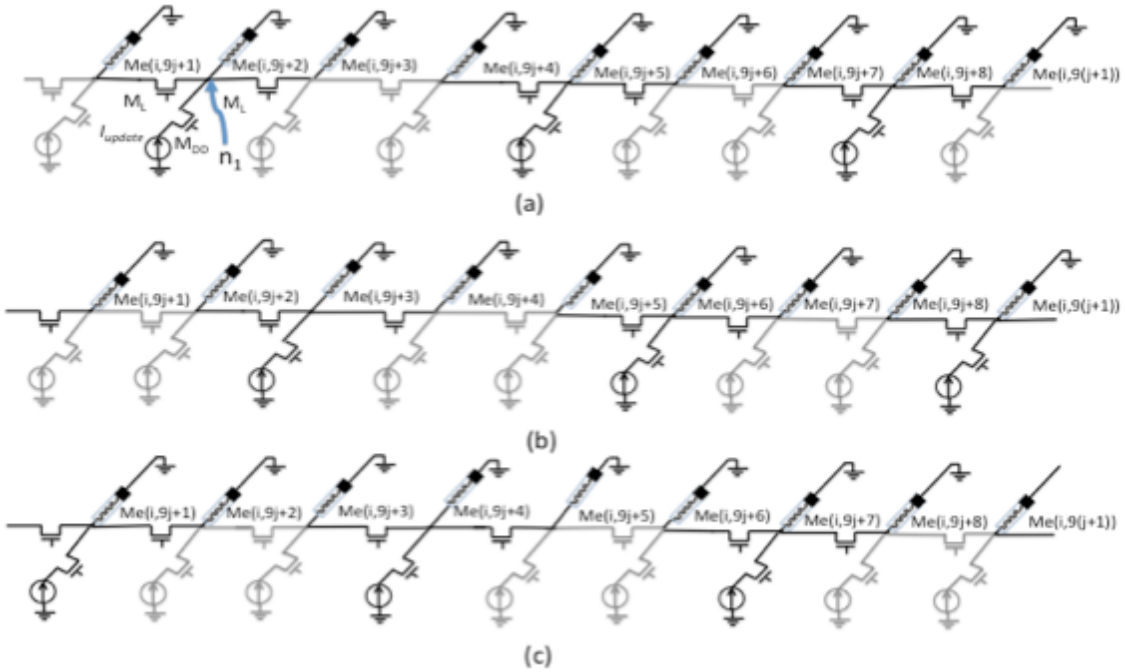


Fig. 6.11. Illustration of ant traversal simulation for a purely horizontal patter of length *L=3*. (a) Illustration of the first ant traversal phase. (b) Illustration of the second ant traversal phase. (c) Illustration of the third ant traversal phase.

In order to sweep all of the design space, different paths are simulated consecutively for the entire image. Furthermore, we should emphasize that although we consider the same path for each and every pixel, the amount of change in the internal variable of the device depends on the value of each memristive device. Furthermore, since this value mimics the pheromone deposit, the amount of pheromone laid on each pixel is different and depends on the location of the pixel.

6.3.5. Stopping criterion, read-out and reset

The stopping criterion is reached once a certain number of ant traversals have been performed. The number of traversal updates is defined by trial and error and the desired quality of the detected edges.

Once the stopping criterion is reached, the conductance of each memristor representing each pixel should be sensed. Activating the read circuitry performs the sensing of the resistance and the final read-out. For this purpose, the word-line (*WL*) on each line is

activated and the bit-line (*BL*) is pre-charged to a small voltage and the *SL* is grounded. Eventually, the *BL* is sensed using a current sense amplifier and the edges are derived.

Finally, once the values of the memristive devices are read out, there is a need to reset all of the devices to ensure correct analysis of consecutive images. For this purpose, a voltage is applied to the BL, SL is grounded and the WL is activated. The voltage causes the memristive device to be reset to its original OFF state.

6.4. Simulation framework for memristive implementation

A simulation framework was developed to investigate edge detection using memristive networks based on swarm intelligence.

The simulation framework consists of four main modules: the memristive device simulation module, the initialization simulation module, the ant traversal simulation module and the read-out/reset module.

6.4.1. Memristive device simulation module

At first, the memristive device simulation module was developed. In order to have a realistic analysis of the algorithm, there was a need to choose a memristive device. There are several memristive devices proposed in literature, e.g. [11,12]. Each of these devices has various characteristics that make them suitable for different applications.

There are several different issues that play important roles in defining the devices suitable for our application. The first important factor is the ability to integrate with CMOS.

The second important factor is the conductance of the memristive device. Observe in Fig. 6.11 that the MOS transistors are used as switches to power gate the memristive devices. Furthermore, as explained earlier, the effectiveness of the algorithm depends on the change in the voltage when the conductance of the memristors changes. Therefore, the r_{ds} of the NMOS should be sufficiently smaller than that of memristive devices to ensure correct operation of the algorithm.

The third important factor is the difference between the ON conductance and the OFF conductance of the device. Since we are considering continuous and gradual change in the conductance and the current passing through the network to change in

accordance with the conductance, higher difference between the ON and the OFF state is desired.

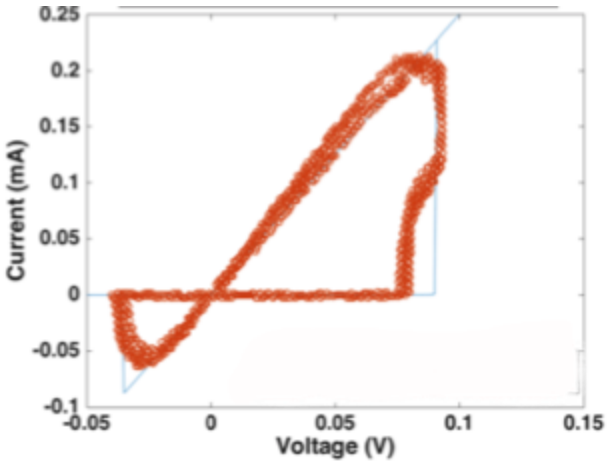
The fourth important factor is the drift constant. The drift constant, defines the rate at which the internal variable changes with respect to the applied voltage. The drift constant plays an important role in the performance and the energy consumption of the network. Larger drift constant results in faster change in the internal variable for a fixed voltage across the device. Therefore, it contributes to the speed of operation. On the other hand, the energy consumption depends on the applied voltage and the time required for each update.

The fifth factor is the relaxation factor. The relaxation factor defines the rate at which the memristive device looses its value, which in turn, corresponds to the evaporation rate of pheromones. In general, the desired relaxation factor depends on the algorithm. Note that not all memristive devices in literature have relaxation factors [81] and this fact should be considered during the design.

The sixth factor, is the type (current based or voltage based) of the memristive device. Due to the perceptible similarities between current based memristive devices and the traversal of ants, they have been used to implement swarm based memristive networks. However, voltage based memristive devices can also be used to implement memristive networks using ant colony as explained in Section 6.3. To this end, ideally, the source transformation of the circuits can be used for realization of the memristive network from one type of device to the other. As an example, current based memristive devices should be connected in series to mimic a path while voltage based memristive elements should be connected in parallel to mimic a path.

Table 6.1. Parameters used to obtain Fig. 6.12 based on the model in [88].

Parameter	Value
Roff	1 MΩ
Ron	400 Ω
Vtp	80e-3
Vtn	-35e-3
βp	19.6e3
βn	17.5e3



Implementation based on [88], Experimental data in [84]

Fig. 6.12. Comparison of model in [88] with the experimental data in [84].

Although theoretically, either of these two types of memristive devices is not preferred over the other, practically, voltage based memristive devices are favorable over current based ones. The main reason for this is the parallel connection of these devices to mimic the ant traversal. The parallel connection prevents stacking of several MOS transistors, used as switches, to ensure their proper operation.

Considering different factors mentioned above, the device in [84] was considered in our work. Furthermore, it was modeled in accordance with the model explained in [88] with different threshold voltages for the positive and negative voltages:

$$f_m = \begin{cases} \beta_p(V_m - V_{tp}) & V_m > V_{tp} \\ -\beta_n(V_m - V_{tn}) & V_m < V_{tn} \\ 0 & V_{tn} < V_m < V_{tp} \end{cases} \tag{6.18}$$

$$\frac{dx}{dt} = f_m, \; R = R_{off}(1 - x) + R_{on}x$$

where V_m is the voltage across the memristor, V_{tp} (V_{tn}) are the positive (negative) threshold voltages and β_p (β_n) are the drift constant for positive (negative) voltages. Also, R_{off} is the resistance of the memristors in the off state and R_{on} is its resistance at the on state. Finally, R is the resistance of the device and x is its internal variable. The model was evaluated in MATLAB and the results were compared against the experimental data in [84]. Fig. 6.12 compares the results obtained by the model in [88] with the data

published in [84]. As observed, the simulation results of our model are in close agreement with the experimental data. The parameters used to obtain these results are shown in Table 6.1.

The memristive device simulation module is used in all the other modules explained in Subsections 6.4.2, 6.4.3, 6.4.4 for transient simulation of the circuit. To this end, the entire circuit is considered with respect to the memristor and the differential equation in Equation 6.18 is solved self consistently. Specifically, at each time step of the transient simulation, the resistance of the memristor is derived based on the current internal variable and the connections in the circuit. Eventually, the voltage and current of each component in the circuit is derived. Furthermore, the value of the internal variable is updated based on the voltage of the memristor derived at each time step.

6.4.2. Initialization circuitry module

The initialization circuitry is responsible to initialize the memristive devices to the contrast of each pixel based on Equation 6.2. Changing the state of the internal variable of the memristive device requires applying a voltage to the device for a certain amount of time. Changing either the voltage or the amount of time the voltage is applied to the memristive device could potentially impact the internal variable of the device. Therefore, the values of the contrast of each pixel could be encoded into the voltage or the amount of time the initialization takes place. Our research and analysis shows that changing the latter is more efficient in terms of energy consumption and performance. Therefore, we chose to use the time entity to encode the value of the initial state.

The initialization circuit takes the value of each pixel as a current source and generates pulses that enable the gating circuit for a certain amount of time. This amount of time depends on the contrast of each pixel. Furthermore, the gating circuit is connected to a fixed voltage that is used to change the value of the internal variable of the memristor.

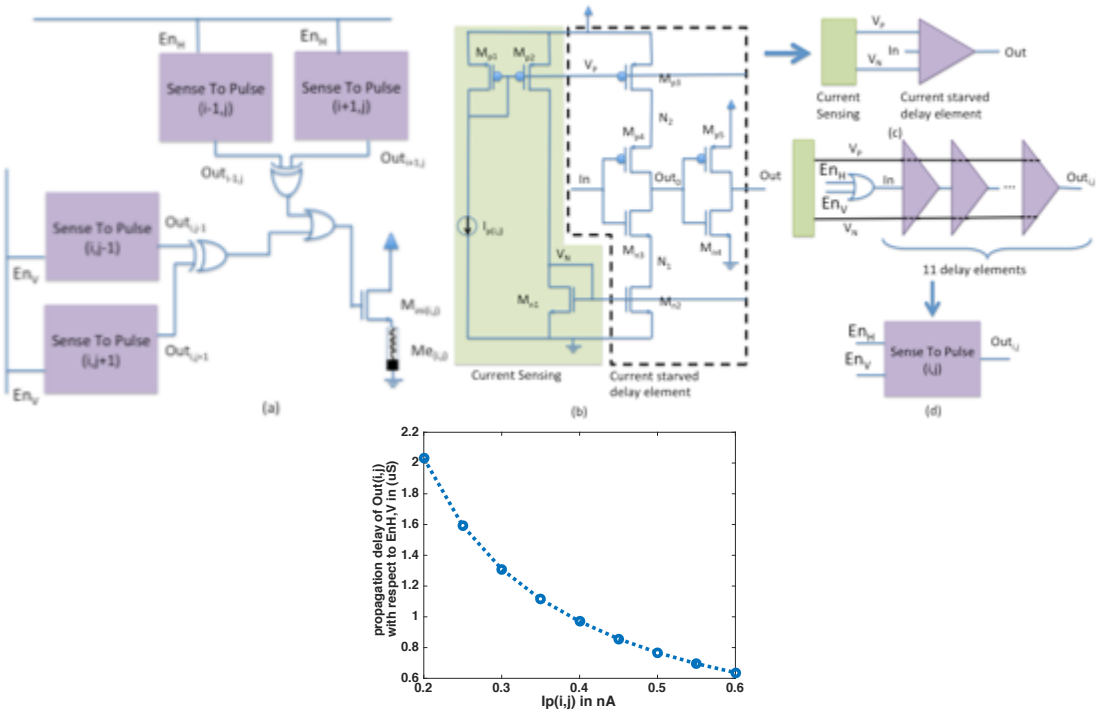


Fig. 6.13. Illustration of the initialization circuit. (a) Circuit connections of the initialization circuit. (b) Current sensing circuit to current starved delay element connection. (c) Symbolic representation of the circuit in (b). (d) Illustration of each pixel's Sense to Pulse circuit. (e) Propagation delay of $Out_{i,j}$ with respect to $En_{H,V}$ vs. $Ip(i,j)$.

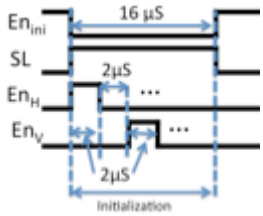


Fig. 6.14. Timing diagram of the control signals for initialization.

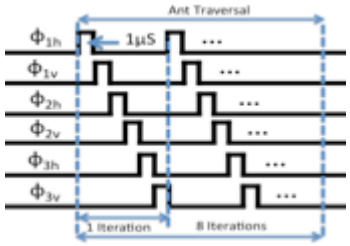


Fig. 6.15. Timing diagram of the control signals for ant traversal.

Fig. 6.13 illustrates the initialization circuit. The initialization is enabled when $\overline{En_{ini}}$ is pulled down. The initialization circuit consists of a single "Current to Pulse" module for each pixel. This circuit generates a pulse with a propagation delay proportional to the value of each pixel. In order to derive the contrast of each pixel, the pulses of the two adjacent pixels are XORed. Therefore, the output of the XOR gates are equal to "1" only when the propagation delays of the "Current to Pulse" modules are different. Furthermore, the initialization is performed in two steps. In the first step, the En_H is enabled and SL is pulled to V_{dd}. Therefore, the contrast of the pixels located horizontally adjacent to the pixel are calculated and fed into the gating circuit. Note that when the pixel values are the same, M_{ini} is ON during the entire En_H. However, if the value of the pixels are different, then M_{ini} is turned off. In the second step, En_V is enabled and SL is pulled up to V_{dd}. Therefore, the contrast of the pixels located vertically adjacent to the pixel are calculated and fed into the gating circuit. Furthermore, we shall point out that in this step, the memristive devices start from the OFF state and end up in a completely ON state if there is no contrast, and to a value in between ON and OFF state if there exists a contrast.

Each "Sense to pulse" module consists of a current sensing module and several current starved delay elements [18].

Table 6.2. Simulation results of the initialization circuitry module.

Parameter	Value
Area	24.356 μm^2
V_{dd}	1.05 V
Power	1 μW ~ 22 μW
Duration of each programming pulse (En_H,En_V)	2 μS
Number of pulses each direction	2
Energy consumption for each programming pulse	6 pJ ~ 132 pJ
$I_{p(i,j)}$	50 pA ~ 1nA

Fig. 6.13 (b) illustrates the connection between these two parts of the circuit. The current source $I_{p(i,j)}$ is considered to have a current proportional to the value of pixel *(i,j)*. Observe that the M_{p1} is diode connected to the current source. Therefore, if the transistors are sized correctly, M_{p2} copies $I_{p(i,j)}$ into its source. Note, M_{n1} is biased with the same current as $I_{p(i,j)}$. This configuration results in changes in V_p and V_n based on the value of $I_{p(i,j)}$. On the other hand, M_{p3} and M_{n2} are used to power the inverter consisting M_{p4} and

M_{n3}. Therefore, Out_0 inverts the signal fed into *In* to Out_0 with a delay proportional to $I_{p(i,j)}$. Finally, the inverter consisting of M_{n4} and M_{p5} is used to stabilize Out_0. Furthermore, if we consider that the delay of this inverter is negligible compared to the current starved inverter, we can consider that *Out* follows *In* with a delay proportional to $I_{p(i,j)}$. Fig. 6.13 (c) illustrates a symbolic representation of the circuit in Fig. 6.13 (b).

In order to increase the delay between the input and the output, several current starved delay elements should be cascaded. Fig. 6.13 (d) illustrates the "Sense to Pulse" module based on these elements. The "Sense to Pulse" module contains an OR gate to enable the input of the delay line with either horizontal (En_H) or vertical (En_V) enable signals. Furthermore, the output will follow the enable signal with a delay proportional to the value of each pixel.

In order to initialize the circuit, the En_{ini} signal is activated and the *SL* is pulled up. At the next step, the En_H and En_V are activated twice as illustrated in Fig. 6.14. Table 6.2 shows the simulation results for the initialization circuit. Note that the control circuitry of initialization consumes less than 10% of the total energy consumption and most of the energy is consumed by the memristive device for changing its internal state.

6.4.3. Ant traversal simulation

The ant traversal circuitry consists of the memristive device together with the transistors used as switches as well as current sources that are used to update the internal variable of the memristive device. In our simulation, we implemented horizontal and vertical patterns only. As explained earlier in Section 6.3, limiting the patterns does not have a crucial effect on the results as long as the length is set properly. Therefore, in our simulation we considered a length of $L=3$ for the ant traversal. The ant traversal was simulated similar to what was explained in Section 6.3 with some modifications. In Section 6.3, we considered the MOS transistors as ideal switches. Therefore, we did not consider the impact of the MOS parameters on the correctness of the implementation. Specifically, we did not consider the impact of the MOS parameters on the implementation. As an example, let us consider the connections in Fig. 6.11. Observe in Fig. 6.11 (a) that *Me(i,9j+2)* is connected to I_{update} through M_{DD} only; however, *Me(i,9j+1)* and *Me(i,9j+3)* are connected to I_{update} through the series of two transistors

M_{DD} and M_L. Therefore, the three memristive devices (*Me(i,9j+1)*, *Me(i,9j+2)*, *Me(i,9j+3)*) are not equal with respect to I_{update}. In other words, if the KCL equation is written for node n_1, we have:

$$I_{update} = \frac{V_{n_1}}{R_{Me(i,9j+2)}} + \frac{V_{n_1}}{R_{Me(i,9j+1)}+R_{dsM_L}} + \frac{V_{n_1}}{R_{Me(i,9j+3)}+R_{dsM_L}} \quad (6.17)$$

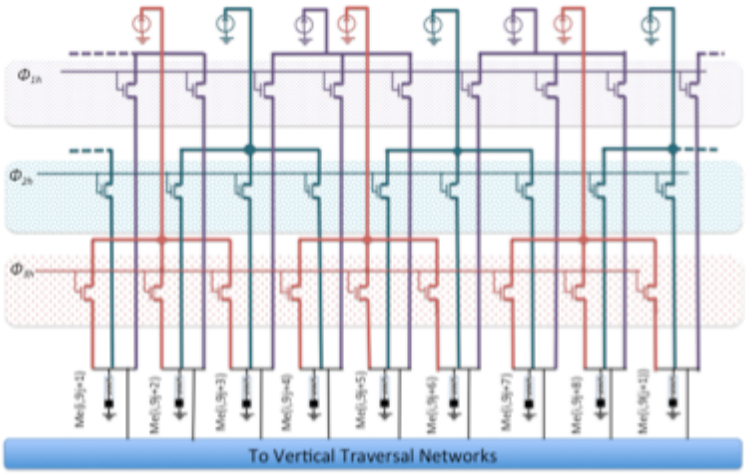


Fig. 6.16. Illustration of the ant traversal update circuitry for purely horizontal pattern of length $L=3$.

Table 6.3. Simulation results of the ant traversal simulation module

Parameter	Value
Area	8.2 μm^2
V_{DD}	1.05 V
Power	3.6 ~6 μW
Duration of each update pulse (ϕ_{ih} or ϕ_{iv})	1 μS
Energy consumption for each update pulse	9 ~ 15 pJ
I_{update}	6 μA

Table 6.4. Simulation results of the read-out/reset module

Parameter	Value
Area	1.54 μm^2
Read V_{DD}	0.5 V
Read power	0.09~1.5 μW
Duration of read	5 nS
Read energy	4.5 ~ 75 fJ
Reset I_{RESET}	4.1~ 8 μW
Reset power	132 μS
Duration of reset pulse	205 ~ 400 PJ
Reset Energy	1.54 μm^2

where V_{n_1} is the voltage at node n_1, $R_{Me(x,y)}$ is the resistance of the memristive devices at location *(x,y)*. Also, R_{dsM_L} is the drain source resistance of M_L. Observe in Equation 6.17 that the effective resistance of the two branches of *(i,(9j+1))* and *(i,9j+3)* are different due to the existence of the M_L transistor. This structural mismatch between the two paths causes disproportionate change in the internal variable in the two adjacent pixels. Furthermore, if the number of the memristive devices in the path increases, this mismatch becomes more pronounced.

In order to solve this problem, each distinct path is implemented using unique transistors. This method of connection ensures symmetric connection to all of the memristive devices in the path. Fig. 6.16 shows a sample connection of memristive devices using the symmetric connection system for the length of *L=3*. Observe in Fig. 6.16 that if ϕ_{i1} is enabled, the first three memristive devices are connected to I_{update}. However, if ϕ_{2h} is enabled, the second memristor is connected to the third and the fourth. In order to simulate the ant traversal, signal ϕ_{ih} is enabled followed by ϕ_{iv}. The ant traversal simulation is performed in several iterations. Each iteration consists of activating the six different ant traversal signals as illustrated in Fig. 6.15. Table 6.3 shows the simulation results for the ant traversal simulation module normalized to each pixel.

6.4.4. Read-out/Reset circuitry module

The read/reset circuitry consists of transistors used to read out the state of the memristive device as well as resetting them to the original state.

In order to read the value of the memristive device, the *WL* is pulled up to V_{dd} and the *BL* is pulled up to a small voltage and *SL* is grounded. At the next step, the current is sensed using a current sense amplifier. Note, passing current through the memristive device could potentially change its internal state. Therefore, M_{read} is designed such that the voltage applied to the device would be less than the threshold voltage of the device. Note that the read operation should be performed for each row separately.

In order to reset the device, *WL* is enabled and *BL* is pulled up to V_{dd} and *SL* is grounded. The reset is performed for all of the memristive devices simultaneously. At the

end of this step, the devices are reset back to the minimum conductance state. Table 6.4 shows the simulation results for the Read-out/Reset circuitry module.

6.5. Simulation results

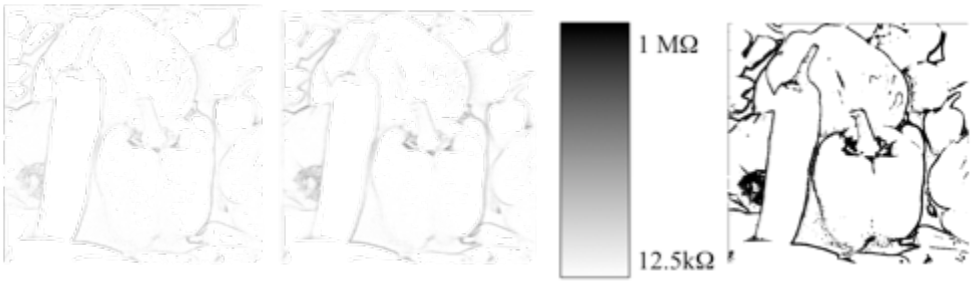
The simulation framework was used to simulate the dynamics of the memristive network. In general, the energy consumption and the termination condition of the algorithm depend on the image. Herein, we provide the results for a case study of the "pepper" image [89]. Fig. 6.17 shows the resistance of the memristors associated with the pixels of the "pepper" image at different simulation times. The number of pixels considered for this implementation is 512x512 pixels. The ON time for each ant traversal was considered to be 1 μs. Our simulations show that the total time required to reach the final state is 60 μs. Furthermore, the energy consumption of the implementation is equal to 0.819 nJ per pixel including the reset energy.

We considered our implementation under non-ideal conditions. As explained in Section 3, bio-inspired algorithms have an inherent immunity to noise. For this purpose, we considered the variations in the form of added noise to the input signal. For this purpose, we added a uniform noise to the value of the pixels. Fig. 6.18 shows the image and the detected edges for different percentage of noise. Observe in Fig. 6.18 that our method generates acceptable results for noise levels up to 30% of the original value.

In order to have a fair comparison with a CMOS implementation, we compared our implementation with some of the state of the art implementations in literature. Recently, it has been shown that image edge detection can be performed using stochastic circuits [90,91] very efficiently. To this end, in [90,91] the authors have simulated custom implementation of these algorithms in hardware. Table 6.5 compares our results with these implementations. Observe in Table 6.5 that our implementation consumes less energy compared to the other two implementations. Furthermore, we would like to emphasize that the data reported in [90,91] does not contain the energy required for the digitization process. It is assumed that the data is already digitized. Therefore, the realistic value of energy consumption for the CMOS implementation is larger than what is reported in [90,91]. However, our implementation is orders of magnitude slower than CMOS. The reason for this poor performance is the slow change in the internal variable

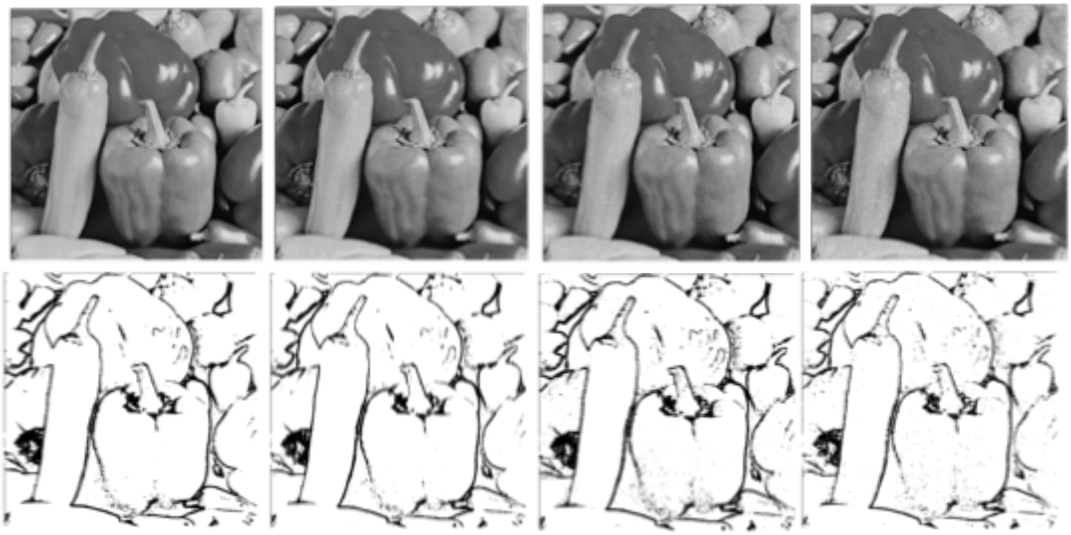
of the memristive devices. We believe that with future advancements in the fabrication of memristive devices, the performance of our methodology can be improved.

(a) original image (b) after initialization (c) iteration =2 (d) iteration =4



(e) iteration = 6(f) iteration = 8 (g) Resistance range (h) extracted edges after read-out

Fig. 6.17. Map of resistance for implementation of "pepper" image at different time samples.



(a) 10% variation (b) 20% variation (c) 30% variation (d) 40% variation

Fig. 6.18. Illustration of image edge detection using the proposed framework for different levels of variations in the intensity values.

Table 6.5. Comparison of image edge detection implementation with CMOS implementations.

Implementation	Area (μm^2)	Delay (μS)	Energy (nJ)
Implementation in [90]	4312	1.3	28.34
Implementation in [91]	200	0.058	1.14
This work	37.22	68	0.819

6.6. Conclusion

In this thesis, we proposed usage of memristive networks for image edge detection based on swarm intelligence. To this end, we proposed a hardware implementation friendly ant colony algorithm for image edge detection. At the next step, we proposed an implementation of the algorithm using memristive devices. Finally, we developed a simulation framework to evaluate our proposed implementation strategy. We implemented the algorithm using state-of-the-art memristive devices. Our results show that our implementation consumes about 5X less area compared to a CMOS implementation of edge detection algorithm. Also, our implementation consumes up to 28% less energy; however, it has three orders of magnitude worse performance. We believe that future advancements in the fabrication of memristive devices could potentially improve the performance of our proposed methodology.

7. CONCLUSION

In this thesis, we explored usage of beyond CMOS devices for various computing purposes. To this end, we explored using Spin Transfer Torque devices for logic implementation as well as enhancing reliability for these devices as memory elements. We also proposed the usage of memristive networks for image edge detection based on swarm intelligence.

First, we proposed a methodology to efficiently implement logic using All Spin Logic (ASL). While ASL devices possess several favorable characteristics such as non-volatility, high density and the ability to operate at low voltages, their suitability to realize arbitrary logic functions remains hitherto unexplored. Towards this end, we proposed an automatic synthesis methodology to design logic circuits using ASL. The design methodology implements three key optimizations *viz.* intra-cycle power gating, nanomagnet stacking and fine-grained logic pipelining, all of which exploit the unique attributes of ASL to improve its energy efficiency. We explored the suitability of ASL for a wide range of benchmarks, including random combinational and sequential logic, DSP data-paths and a general purpose Leon SPARC3 processor and compared ASL to CMOS at the 16nm technology node. We identify that the delay or the switching speed of ASL nanomagnets together with the short spin diffusion length of the non-magnetic channels are key bottlenecks to their efficiency. Specifically, the short spin diffusion length requires insertion of interconnect buffers resulting in significant degradation on performance and energy consumption.

On the other hand, logic optimizations such as fine-grained pipelining are targeted to improve the energy consumption, they may not be possible in all cases due to the presence of sequential dependencies in the logic. In such cases, the current required to match the performance of CMOS can be quite high, and this significantly impacts the energy-efficiency of ASL. Other optimizations such as nanomagnet stacking also

significantly improve the efficiency of ASL. However, the degree of stacking or the number of nanomagnets that can be stacked together is constrained by the physical limitations of routing.

We also considered improving reliability of STT-MRAM memory arrays by considering a broader design space that covers different levels of abstraction. To this end, we proposed a device/circuit/architecture simulation framework and co-design methodology for STT-MRAM. Our simulation framework consists of an MTJ device level simulator, a bit-cell failure analysis model, and an array level ECC simulator. We first calibrated our simulation framework to device characteristics that were experimentally obtained and published in the literature before using it to evaluate a 64MB STT-MRAM array with 64-bit word length. The memory array was optimized by varying bit-cell parameters in conjunction with different ECC schemes. We found that the conventional design methodology where the bit-cell and array level designs are separately optimized yield sub-optimal designs. We found that significant improvements to the array design may be obtained using our proposed device/circuit/array co-design methodology. At iso-array area, our methodology achieved two orders of magnitude reduction in total probability of failure. Moreover, our methodology can achieve 13% smaller array area at the same total array probability of failure. We believe that device/circuit/array co-design is crucial for achieving dense, energy efficient and robust STT-MRAM arrays.

Subsequently, we analyzed the impact of process variations on the run-time reliability of STT-MRAM memory arrays. Furthermore, we analyzed the efficacy of ECC in relaxing E_B requirement of the MTJ under process variations. We also analyzed the efficacy of ECC on yield enhancement and run-time reliability. Our results showed that if SECDED is used for yield enhancement besides run-time reliability, it may be difficult to have a more relaxed value of E_B (better write current). Thus, we proposed using FaECC in which permanent faults are masked using SECDED while maintaining its correction capability for soft errors. In order to analyze the efficacy of FaECC, we performed a case study of a 1 MB cache in 32 nm Technology node. We showed that in our proposed scheme, the area of the memory array is reduced up to 20% compared to a cache with

SECDED and up to 13% compared to a cache with DECTED, at iso-reliability. Furthermore, the write energy can be reduced up to 21% and 11% compared to caches with SECDED and DECTED correction capabilities, respectively.

Eventually, we proposed memristive networks for image edge detection based on swarm intelligence. To this end, we proposed a hardware implementation friendly ant colony algorithm for image edge detection. We developed a simulation framework to evaluate our proposed implementation strategy. To this end, we implemented the algorithm using state-of-the-art memristive devices. Our results show that our implementation consumes about 5X less area compared to a hardware implementation of image edge detection using stochastic circuits. Also, our implementation consumes up to 28% less energy; however, it has three orders of magnitude worse performance. We believe that future advancements in the fabrication of memristive devices could potentially improve the performance of our proposed methodology.

LIST OF REFERENCES

LIST OF REFERENCES

[1] International Technology Roadmap for Semiconductors, Chapter PIDS, 2011. [Online]. Available: http://www.itrs.net/

[2] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff, ‘‘Limits to binary logic switch scaling -A Gedanken model,’’ Proc. IEEE, vol. 91, no. 11, pp. 1934–1939, Nov. 2003.

[3] J.C. Slonczewski, “Current-driven excitations of magnetic multilayers,” J. Magn. Magn. Mater. 159 (1996) L1.

[4] L. Berger, “Emission of spin waves by a magnetic multilayer traversed by a current,” Phys. Rev. B 54 (1996) 9353; L. Berger, J. Appl. Phys. 90 (2001) 4632.

[5] D. C. Ralph, M. D. Stiles, “Current Perspectives: Spin transfer torques,” in J. Magn. Magn. Mater. 320 (2008) 1190-1216.

[6] Gallagher, William Joseph, et al. "Magnetic memory array using magnetic tunnel junction devices in the memory cells." U.S. Patent No. 5,640,343. 17 Jun. 1997.

[7] Jedema, F. J., A. T. Filip, and B. J. Van Wees. "Electrical spin injection and accumulation at room temperature in an all-metal mesoscopic spin valve."*Nature* 410.6826 (2001): 345-348.

[8] Liu, Luqiao, et al. "Spin-torque switching with the giant spin Hall effect of tantalum." *Science* 336.6081 (2012): 555-558.

[9] B. Del Bel, J. Kim, C. H. Kim, S. S. Sapatnekar, “ Improving STT-MRAM density through multibit error correction,” in DATE 2014.

[10] J. Li, C. Augustine, S. Sahahuddin, K. Roy,” Modeling of failure probability and statistical design of spin-torque transfer magnetic random access memory (STTMRAM) array for yield enhancement,” in DAC 2008, 45th ACM/IEEE.

[11] J. Li, H. Liu, S. Salahuddin, K. Roy,”Variation-tolerant spin-torque transfer (STT) MRAM array for yield enhancement,” in IEEE 2008 Custom Integrated Circuits Conference (CICC).

[12] J. Li, P. Ndai, A. Goel, S. Salahuddin, K. Roy," Design paradigm for robust spin-torque transfer magnetic RAM (STTMRAM) from circuit/architecture perspective," in IEEE Trans. On VLSI syst., vol 18, No.12, Dec. 2010.

[13] X. Fong, S. H. Choday, K. Roy," Bit-cell level optimization for non-volatile memories using magnetic tunnel junctions and spin transfer torque switching," in IEEE Trans. on Nanotechnology, vol. 11, No. 1 Jan. 2012.

[14] Y.J. Lee, G. Jan, Y. J. Wang, K. Pi, " Demonstration of chip level writability, endurance and data retention of an entire 8Mb STT-MRAM array," VLSI Technology, Syst. And App., Intl. Symp. On, April 2013.

[15] W. Xu, Y. Chen, X. Wang, T. Zhang, "Improving STT MRAM storage density through smaller-than-worst-case transistor sizing." DAC 2009.

[16] W. Xu, H. Sun, X. Wang, Y. Chen, " Design of last level on-chip cache using spin torque transfer RAM (STT RAM)," in IEEE Trans. on VLSL vol. 19, no. 3, Mar. 2011.

[17] Sharad, Mrigank, et al. "Design of ultra high density and low power computational blocks using nano-magnets." *Quality Electronic Design (ISQED), 2013 14th International Symposium on*. IEEE, 2013.

[18] A. Sarkar, D. E. Nikonov, I. A. Youngh, B. Behin-Aein, S. Datta, " Charge-resistance approach to benchmarking performance of beyond-CMOS information processing devices," in IEEE Trans. on Nanotechnology, Vol. 13, No. 1, pp. 143-150, Jan. 2014.

[19] K. Bernstein, R. K. Cavin, W. Porod, A. Seabaugh, " Device and architecture outlook for beyond CMOS switches," Proc. IEEE, vol. 98, no. 12, pp. 2169-2184, Dec. 2010.

[20] T. Kimura, Y. Otani, and J. Hamrle.," Switching magnetization of a nanoscale ferromagnetic particle using nonlocal spin injection," Phys. Rev. Lett.,96,037201, Jan 2006.

[21] J. Sun, M. Gaidis, E. OSullivan, E. Joseph, et al, "A three-terminal spin-torque-driven magnetic switch," Applied physics letters, 95(8):083506, 2009.

[22] C. F. Pai, L. Liu, Y. Li, H. W. Tseng, D. C. Ralph, R. A. Buhrman, " Spin transfer torque devices utilizing the spin Hall effect of tungsten," in Bulletin of the American Physical Society, vol. 58, no.1, 2013.

[23] H. Liu, D. Bedau, D. Backes, J. A. Katine, J. Langer, A. D. Kent, " Ultrafast switching in magnetic tunnel junction based orthogonal spin transfer devices," in Applied Physics Letters, Lett. 97, 2010.

[24] B. Behin-Aein, A. Sarkar, S. Srinivasan, and S. Datta, "Switching energy-delay of all spin logic devices," Applied Physics Letters, 98(12):123510-123510, 2011.

[25] S. Srinivasan, V. Diep, B. Behin Aein, A. Sarkar, S. Datta," Modeling multi-magnet networks interacting via spin currents," in www.arxiv.org/abs/1304.072.

[26] A. Sarkar, S. Srinivasan, B. Behin-Aein, and S. Datta, "Modeling all spin logic: Multi-magnet networks interacting via spin currents. In Electron Devices Meeting," (IEDM), 2011 IEEE International, pages 1-11. IEEE, 2011.

[27] C. Augustine, G. Panagopoulos, B. Behin-Aein, S. Srinivasan, A. Sarkar, and K. Roy, "Low-power functionality enhanced computation architecture using spin-based devices," In Nanoscale Architectures (NANOARCH), 2011 IEEE/ACM International Symposium on, pages 129-136. IEEE, 2011.

[28] A. Brataas, Y.V. Nazarov, G.E.W. Bauer, Finite element theory of transport in ferromagnet-normal metal systems, Phys. Rev. Lett. 84, (2000) 2481–2484.

[29] S. Manipatruni, D. E. Nikonov, I. A. Young, " Circuit theory for SPICE of Spintronic Integrated Circuits," in www.arxiv.org 2011arXiv1112.2746M.

[30] T. Valet and A. Fert, "Theory of the perpendicular magnetoresistance in magnetic multilayers", Phys. Rev. B 48, 7099 (1993).

[31] L. He, W. D. Doyle, and H. Fujiwara, "High-speed switching in magnetic recording media," IEEE Trans. Magn. 30, 4086 ~1994.

[32] S. Salahuddin, and S. Datta, "Self-consistent simulation of quantum transport and magnetization dynamics in spin-torque based devices," Applied Physics Letters, 15, 2006, p. 153504.

[33] Design Compiler, Synopsys Inc.

[34] J. M. Renders, H. Bersini, " Hybridizng genetic algorithms with hill-clobiming methods for global optimization: two possible ways," in Proc. Of first IEEE World Congress on Computational Intelligence, Jun. 1994.

[35] S. H. Gerez, *Algorithms for VLSI Design Automation,* John Wiley, 1999.

[36] I. Ahmad, Y. K. Kwok, " On exploiting task duplication in parallel program scheduling," in IEEE Trans. Parallel Distrib. Systems, vol. 9, pp. 872-892, 1998.

[37]T. Yang, A. Gerasoulis, " List scheduling with and without communication delays," in Parallel Computing, Elsevier, 1993.

[38] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," in IEDM 2012.

[39] G. Karypis, V. Kumar, "hMetis 1.5: A hypergraph partitioning package," tech. report, Univ. Minnesota, 1998.

[40] G. Karypis, V. Kumar, " hMetis: A Hypergraph Partitioning Package Version 1.5," user manual, 1998.

[41] A. E. Caldwell, A. B. Kahng, I. L. Markov, "Can recursive bisection alone produce routable placements?," DAC, 2000.

[42] J. Bass, W. P. Partt Jr.,"Spin-Diffusion Lengths in Metals and Alloys, and Spin-Flipping at Metal/Metal Interfaces: an Experimentalist's Critical Review," in J Phys.: Condens. Matter 19, 2007.

[43] Nishimura, Naoki, et al. "Magnetic tunnel junction device with perpendicular magnetization films for high-density magnetic random access memory *Journal of applied physics* 91.8 (2002): 5246-5249.

[44] Sun, J. Z. "Spin-current interaction with a monodomain magnetic body: A model study." *Physical Review B* 62.1 (2000): 570.

[45] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0", Microelectronics Center of North Carolina (MCNC), 1991.

[46] Leon SPARC3 Processor,www.gaisler.com/ index.php/ products/ processors/ leon3

[47] S. Sinha, G. Yeric, V. Chandra, B. Cline, Y. Cao, "Exploring sub-20nm FinFET design with predictive technology models," Proc. DAC, 2012.

[48] P. Bruski, Y. Manzke, R. Farshchi, O. Brandt, J. Herfort, M. Ramsteiner, "All-electrical spin injection and detection in the Co2FeSi/GaAs hybrid system in the local and non-local configuration," Appl. Phys. Lett., 103, 052406 (2013).

[49] T. Shima, K. Takanashi, Y. K. Takahashi, K. Hono, J., " Nucleation-type magnetization behavior in FePt (001) particulate films," Appl. Phys. 10N122 (2005).

[50] Drögeler, Marc, et al. "Nanosecond spin lifetimes in single-and few-layer graphene-hBN heterostructures at room temperature," *Nano letters* (2014).

[51] X. Wang, Y. Chen, H. Xi, H. Li, and D. Dimitrov, "Spintronic memristor through spin-torque-induced magnetization motion," *IEEE Elect. Dev. Lett.* vol. 30, no. 3, pp. 294-297, Mar 2009.

[52] Y. Huai, F. Albert, P. Nguyen, M. Pakala, and T. Valet, "Observation of spin-transfer switching in deep submicron-sized and low-resistance magnetic tunnel junctions," *Appl. Phys. Lett.* vol. 84, no. 16, pp. 3118-3120, Apr 2004.

[53] S. Yuasa, T. Nagahama, A. Fukujima, Y. Suzuki, and K. Ando, "Giant room-temperature magnetoresistance in single-crystal Fe/MgO/Fe magnetic tunnel junctions," *Nature Materials* vol. 3, pp. 868-871, Dec 2004.

[54] P. Krzysteczko, X. Kou, K. Rott, A. Thomas, and G. Reiss, "Current induced resistance change of magnetic tunnel junctions with ultra-thin MgO tunnel barriers," *J. Magn. Magn. Mater.* vol. 321, 3, pp. 144-147, Feb 2009.

[55] S. Salahuddin and S. Datta, "Spin transfer torque as a non-conservative pseudo-field," arXiv:0811.3472.

[56] T. Kawahara, R. Takemura, K. Miura, J. Hayakawa, S. Ikeda, Y. M. Lee, R. Sasaki, Y. Goto, K. Ito, T. Meguro, F. Matsukura, H. Takahashi, H. Matsuoka, and H. Ohno, "2 Mb SPRAM (SPin-transfer torque RAM) with bit-by-bit bi-directional current write and parallelizing-direction current read," in *Proc. IEEE Int. Solid-State Circuits '07*, pp. 480-482, Feb 2007.

[57] C. J. Lin, S. H. Wang, Y. J. Wang, K. Lee, X. Zhu, W. C. Chen, X. Li, W. N. Hsu, Y. C. Kao, M. T. Liu, W. C. Chen, Y. Lin, M. Novak, N. Yu and L. Tran, "45nm low power CMOS logic compatible embedded STT MRAM utilizing a reverse-connection 1T/1MTJ cell," in *Proc. IEEE Int. Electr. Dev. Meeting '09*, pp. 279-282, Dec 2009.

[58] R. Nebashi, N. Sakimura, H. Honjo, S. Saito, Y. Ito, S. Miura, Y. Kato, K. Mori, Y. Ozaki, Y. Kobayashi, N. Ohshima, K. Kinoshita, T. Suzuki, K. Nagahara, N. Ishiwata, K. Suemitsu, S. Fukami, H. Hada, T. Sugibayashi, and N. Kasai, "A 90nm 12ns 32Mb 2T1MTJ MRAM," in *Proc. IEEE Int. Solid-State Circuits '09*, pp. 462-463, 2009.

[59] Y. Huai, "Spin-transfer torque MRAM (STT-MRAM): challenges and prospects," *AAPS Bulletin* vol. 18, no. 6, pp. 33-40, Dec 2008.

[60] X. Yao, H. Meng, Y. Zhang, and J. Wang, "Improved current switching symmetry of magnetic tunneling junction and giant magnetoresistance devices with nano-current-channel structure," *J. Appl. Phys.* vol. 103, 7, pp. 07A717-1-07A717-3, Mar 2008.

[61] X. Fong, S. H. Choday, K. Roy," Bit-cell level optimization for non-volatile memories using magnetic tunnel junctions and spin transfer torque switching," in IEEE Trans. on Nanotechnology, vol. 11, No. 1 Jan. 2012.

[62] V. K. Wei, et. al., "On the generalized Hamming weights of product codes." *IEEE transactions on information theory* 39, no. 5 (1993): 1709-1713.

[63] T. Kasami, "Weight distribution formula for some class of cyclic codes ," Vol. 285. Illinois Univ. Coordinated Science Lab, 1966.

[64] Z. Pajouhi, X. Fong, K. Roy, " Device/Circuit/Architecture co-design of reliable STT-MRAM, " to appear in DATE 2015.

[65] X. Fong, et. al., "KNACK: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque MRAM bit-cells," in SISPAD 2011.

[66] https://nanohub.org/resources/21259

[67]http://www.synopsys.com/tools/verification/amsverification/circuitsimulation/hspice/Pages/default.aspx

[68] A. Raychowdhury, et. al., " Design Space and Scalability Exploration of 1T-1STT MTJ memory arrays in the presence of variability and disturbances," IEDM 2009.

[69]http://www.synopsys.com/COMMUNITY/UNIVERSITYPROGRAM/Pages/32-28nm-generic-library.aspx

[70] C. Augustine, A. Raychowdhury, D. Somasekhar, J. Tschanz, K. Roy, V. K. De, " Numerical analysis of typical STT_MTJ stacks for 1T-1R memory arrays," in IEDM 2010.

[71] D. Apalkov, Z. Diao, A. Panchula, S. Wang, Y. Huai, and K. Kawabata, "Temperature dependence of spin transfer switching in nanosecond regime," IEEE Trans. Magn., vol. 42, no. 10, pp. 2685–2687, Oct. 2006.

[72] C.Wilkerson, A.R.Alameldeen, Z.Chishti, W.Wu, D.Somasekhar,and S. L. Lu, "Reducing Cache Power with Low-Cost, Multi-bit Error-Correcting Codes," in *Proc. ISCA'10*, Jun. 2010, pp. 83-93.

[73] D. Strukov, "The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories," in *Proc. ACSSC*, Oct. 2006, pp. 1183-1187.

[74] S. Evain, V. Savin, V. Gherman, " Error correction schemes with erasure information for fast memories," in Springer Journal of electron test, vol. 30, pp. 185-192, 2014.

[75] C. E. Walker, C.-E. Sundberg, and C. Black, "A reliable spaceborne memory with a single error and erasure correction scheme," *Computers, IEEE Transactions on*, vol. C-28, no. 7, pp. 493–500, 7 1979.

[76] C. L. Chen, and M. Y. Hsiao. "Error-correcting codes for semiconductor memory applications: A state-of- the-art review." *IBM Journal of Research and Development* 28, no. 2 (1984): 124-134.

[77] K. Kwon, X. Fong, P. Wijesinghe, P. Panda, K. Roy," High density and robust STT-MRAM array through device/circuit/architecture interactions," in IEEE Trans. Nanotechnology, volume PP, Issue 99, early access, (http://ieeexplore.ieee.org/xpl/abstractAuthors.jsp?arnumber=7159082&filter%3DAND(p_IS_Number%3A435 9107)) accessed on 31/7/2015.

[78] A. Sengupta, Y. Shim, and K. Roy, "Simulation studies of an All-Spin Artificial Neural Network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets", arXiv:1510.00459.

[79] Jo, Sung Hyun, et al. "Nanoscale memristor device as synapse in neuromorphic systems." *Nano letters* 10.4 (2010): 1297-1301.

[80] http://phys.org/news/2013-04-energy-efficient-brain-simulator-outperforms.html

[81] Y. V. Pershin and M. Di Ventra, “Solving mazes with memristors: A massively-parallel approach,” Phys. Rev. E, vol. 84, p. 046703, 2011.

[82] Y. V. Pershin and M. Di Ventra, "Memcomputing and Swarm Intelligence." *arXiv preprint arXiv:1408.6741* (2014).

[83] M. Dorigo, V. Maniezzo and A. Colorni "Ant System: Optimization by a Colony of Cooperating Agents", IEEE Trans, Systems, Man and Cybernetics, Part B, vol. 26, no. 1, pp.29 -41 1996.

[84] Hasegawa, Tsuyoshi, et al. "Atomic Switch: Atom/Ion Movement Controlled Devices for Beyond Von-Neumann Computers." *Advanced Materials* 24.2 (2012): 252-267.

[85] H. Nezamabadi-pour, et. al., "Edge detection using ant algorithms." *Soft Computing* 10.7 (2006): 623-628.

[86] J. Tian, W. Yu, and S. Xie, “An ant colony optimization algorithm for image edge detection,” IEEE Congress on Evolutionary Computation, pages 751-756, 2008.

[87] S. A. Etemad and T. White "An ant-inspired algorithm for detection of image edge features", Applied Soft Computing, vol. 11, no. 8, pp.4883 - 4893, 2011.

[88] Biolek, Dalibor, Massimiliano Di Ventra, and Yuriy V. Pershin. "Reliable SPICE simulations of memristors, memcapacitors and meminductors." *arXiv preprint arXiv:1307.2717* (2013).

[89] http://sipi.usc.edu/database/database.php?volume=misc

[90] Li, Peng, et al. "Computation on stochastic bit streams digital image processing case studies." *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 22.3 (2014): 449-462.

[91] Alaghi, Armin, Cheng Li, and John P. Hayes. "Stochastic circuits for real-time image-processing applications." *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013.

VITA

VITA

Zoha Pajouhi is a Ph.D. candidate at Nanoelectronics Research Laboratory, in ECE at Purdue University, West Lafayette, IN. Her research interests include CAD methodology and reliability analysis for beyond CMOS and spin-based devices. Also, she is interested in non-boolean and bio-inspired computing using beyond CMOS devices. Previously, she has also worked on VLSI design for signal processing and communication systems.

Zoha received her B.S. and M.S. in electrical engineering from Sharif University of Technology and University of Tehran respectively.